



Escuela  
Politécnica  
Superior

# Aplicación móvil enfocada al turismo en ciudades



Grado en Ingeniería Multimedia

## Trabajo Fin de Grado

Autor:

Alejandro Martínez Martínez

Tutor/es:

Javier Montoyo Bojo

Septiembre 2018



Universitat d'Alacant  
Universidad de Alicante



## Resumen

En el siguiente proyecto se va a llevar a cabo una aplicación móvil enfocada al turismo y que basa gran parte de su funcionalidad en la interacción de los usuarios sobre la misma. Para ello se llevará a cabo una investigación sobre el turismo y cómo es afectado por las nuevas tecnologías. Una vez realizada la investigación se creará la aplicación en base a los datos obtenidos para, de esa manera, obtener el mejor resultado posible de cara a los posibles clientes.



## Motivación y justificación

Desde que obtuve mi primer ordenador con 9 años y pude sentir de primera mano las posibilidades que la tecnología ofrecía, supe que mi vocación sería la informática. A partir de este momento pasé gran parte de mi tiempo libre jugando a videojuegos o interactuando con herramientas que me permitían realizar todo tipo de trabajos web, desde la creación de pequeños portales a sencillas aplicaciones. Todo este interés fue creciendo a medida que pasaban los años. Ya no sólo jugaba a videojuegos, sino que dedicaba parte del tiempo a investigar cómo esos juegos se llevaban a cabo.

Todo ese gusto por la informática fue total en el momento que obtuve mi primer dispositivo táctil, en este caso, un teléfono móvil. Quedé tan sorprendido con ese tipo de tecnología que sabía que en un futuro terminaría trabajando en algo relacionado con la creación de contenido para ese tipo de dispositivos. Y así ha sido, siempre que hemos tenido la oportunidad de realizar algún trabajo libre durante el transcurso de la carrera, decantaba mis proyectos a los dispositivos portátiles. Ya fuesen aplicaciones móviles, sencillos juegos o páginas web adaptadas a estos.

Los años fueron pasando hasta que llegó cuarto curso, y con él, el Trabajo de Fin de Grado (TFG) y el proyecto de Aprendizaje Basado en Proyectos (ABP). Estos dos proyectos me han dado la oportunidad de sumergirme más aún en el mundo de los dispositivos móviles y con la ayuda de mi tutor Javier Montoyo decidí, después de unas cuantas reuniones, que una opción interesante era la creación de una aplicación interactiva que ayude a todos los usuarios que realicen un viaje.

Con este proyecto pretendo aumentar mis conocimientos sobre tecnologías enfocadas al desarrollo de aplicaciones móviles y, sobre todo, tener la opción de crear algo que pueda ser de ayuda para la gente además de ser llamativo, funcional y útil para el mayor número posible de personas.



## Agradecimientos

A todos los profesores que durante estos años de carrera han hecho que me interese aún más en el mundo de la tecnología y sobre todo a mi tutor Javier por ayudarme y guiarme en la realización de este proyecto.





*No es que usemos la tecnología, vivimos con la tecnología.*

*Godfrey Reggio.*



# Índice de Contenidos

<b>1. INTRODUCCIÓN.....</b>	<b>15</b>
<b>2. ESTADO DEL ARTE .....</b>	<b>17</b>
2.1. SOLUCIONES ACTUALES .....	17
2.2. TECNOLOGÍAS PARA EL DESARROLLO .....	20
2.2.1. <i>Capa de datos</i> .....	21
2.2.2. <i>Capa de negocio</i> .....	23
2.2.3. <i>Capa de presentación</i> .....	24
<b>3. OBJETIVOS .....</b>	<b>28</b>
<b>4. METODOLOGÍA .....</b>	<b>29</b>
<b>5. ANÁLISIS Y ESPECIFICACIÓN.....</b>	<b>31</b>
5.1. LIENZO DE MODELO DE NEGOCIO (LEAN CANVAS).....	31
5.2. REQUISITOS FUNCIONALES.....	34
5.3. REQUISITOS NO FUNCIONALES.....	36
5.4. TECNOLOGÍAS UTILIZADAS.....	37
<b>6. DISEÑO .....</b>	<b>40</b>
6.1. ARQUITECTURA DE LA APLICACIÓN .....	40
6.2. BASE DE DATOS .....	43
6.3. API .....	45
6.4. INTERFACES .....	47
6.5. GUÍA DE ESTILOS.....	50
<b>7. IMPLEMENTACIÓN .....</b>	<b>52</b>
7.1. BASE DE DATOS .....	52
7.2. SERVIDOR .....	52
7.3. SEGURIDAD EN EL SERVIDOR .....	53
7.4. CLIENTE.....	56
7.5. MANIPULACIÓN DOM .....	57
7.6. TRATAMIENTO DE IMÁGENES .....	58
<b>8. PRUEBAS Y VALIDACIÓN .....</b>	<b>60</b>

8.1.	VALIDACIÓN DE RESULTADOS .....	60
8.2.	PRUEBAS Y RESULTADOS DE USABILIDAD .....	62
<b>9.</b>	<b>RESULTADOS.....</b>	<b>64</b>
9.1.	APLICACIÓN .....	64
9.1.1.	<i>Login y Registro .....</i>	<i>64</i>
9.1.2.	<i>Mapa .....</i>	<i>65</i>
9.1.3.	<i>Añadir Lugar .....</i>	<i>66</i>
9.1.4.	<i>Páginas foro y respuestas.....</i>	<i>67</i>
9.1.5.	<i>Añadir pregunta .....</i>	<i>67</i>
9.1.6.	<i>Añadir Respuesta.....</i>	<i>68</i>
9.1.7.	<i>Perfil .....</i>	<i>69</i>
9.1.8.	<i>Información de un lugar .....</i>	<i>70</i>
<b>10.</b>	<b>CONCLUSIONES .....</b>	<b>71</b>
10.1.	TRABAJO FUTURO .....	71
<b>11.</b>	<b>BIBLIOGRAFÍA Y REFERENCIAS.....</b>	<b>73</b>

# Índice de Figuras

Figura 1: SCRUM .....	30
Figura 2: Lean Canvas.....	34
Figura 3: Arquitectura de la aplicación .....	41
Figura 4: Arquitectura Modelo Vista Controlador .....	43
Figura 5: Esquema de la base de datos.....	45
Figura 6: Boceto Login y Registro.....	48
Figura 7: Bocetos vistas principales.....	49
Figura 8: Bocetos vistas secundarias.....	50
Figura 9: Paleta de Colores Interfaces.....	51
Figura 10: Mapas de la Aplicación .....	61
Figura 11: Páginas Login y registro .....	64
Figura 12: Página de Mapa.....	65
Figura 13: Página Añadir Lugar.....	66
Figura 14: Página foro y respuestas .....	67
Figura 15: Página añadir lugar .....	68
Figura 16: Página añadir respuesta.....	69
Figura 17: Páginas Perfil, Mis lugares, Mis respuestas.....	70
Figura 18: Página Información Lugar y vista previa.....	70

## Índice de Tablas

Tabla 1: Comparación de Bases de Datos .....	22
Tabla 2: Comparación tipos de Aplicaciones.....	25
Tabla 3: API Rest .....	47



# 1. Introducción

Este trabajo consiste en la realización de una aplicación móvil de índole turístico que recoge información de servicios públicos, en este caso de Google Maps [7], para posteriormente mostrársela al usuario. Algunos de los datos que se podrán visualizar desde la aplicación son:

- Descripción.
- Ubicación.
- Ruta para llegar a dicho lugar.
- Fotos.
- Comentarios de otros visitantes.

Además de lo mencionado anteriormente, también contará con una base de datos propia que permitirá a los usuarios registrados poder dar de alta sitios personalizados que no aparezcan en los servicios públicos y que crean relevantes para otros visitantes. Los usuarios tendrán la posibilidad de añadir sitios personalizados introduciendo un nombre, descripción, foto y posición del mismo, permitiéndonos realizar todo esto desde la misma aplicación, incluso la carga y subida de imágenes. A esto le añadimos un sistema de puntuación que permite a los usuarios valorar los distintos sitios de la plataforma para, así, proporcionar información adicional al resto de personas, ya que los usuarios suelen tener muy en cuenta la opinión de otras personas que ya han realizado una visita a este lugar.

Otra opción con la que cuenta la aplicación es un mapa cargado desde Google Maps en el que se incluyen los sitios anteriormente mencionados, diferenciados con distintos colores e iconos dependiendo de la categoría en la que se encuentren. La vista del mapa nos permite cargar una ruta desde nuestra posición hasta el lugar de interés que seleccionemos. Además, también contamos con un buscador en el que introducimos el lugar de inicio y el destino y nos genera la ruta.

La aplicación también cuenta con un foro que es utilizado para que los usuarios que lo necesiten puedan dejar sus preguntas y permitir a otros responderlas y ayudarles. En cualquier momento podremos ver todas las preguntas realizadas sobre un sitio en concreto, gracias a un buscador incluido en la aplicación que filtrará las preguntas realizadas por la ciudad en la que se encuentran, ya que, de manera predeterminada, nos aparecen las últimas que han sido formuladas.

No siempre vamos a querer la información de nuestro alrededor, es posible que algunos usuarios, antes de realizar un viaje, quieran ver lo que se van a encontrar en su destino. Para ello, al entrar en la aplicación, se nos muestra la información alrededor de la posición en la que nos



encontramos, pero tenemos la opción de navegar por todo el mapa para observar los distintos puntos de interés con los que cuentan las diferentes ciudades alrededor del mundo. También podemos filtrar los datos que se nos muestran, de esta forma se podrá ver la información de nuestra base de datos, la devuelta por los servicios o ambas.

## 2. Estado del arte

El turismo está en auge. Esta es una realidad desde hace mucho tiempo, de hecho, según la Organización Mundial del Turismo [19], los resultados obtenidos en 2017 son los más altos en los últimos siete años. Se produjo un aumento del 7% en comparación al año anterior y se estima que en 2018 este aumento se mantenga y siga creciendo a un ritmo de entre un 4% y un 5%.

Este crecimiento, sumado a la aparición de las nuevas tecnologías como las tablets o los smartphones han hecho que muchas personas quieran unir estos dos mundos. De esta manera cada vez más personas quieren incluir el uso cotidiano de dispositivos móviles también en sus vacaciones. A estos usuarios se les llama *turistas digitales*.

Los turistas digitales se diferencian de los convencionales en diversas cosas, algunas de las más importantes son:

- Investiga con antelación, no deja todo el proceso en manos de una agencia. Este tipo de turista revisa comentarios y valoraciones para hacerse una idea de lo que se va a encontrar.
- Quiere vivir experiencias desde el principio, y una buena manera de ello es ver imágenes o vídeos de su destino.

Según “The Valley Digital Business School “[22] las 6 claves para atraer a los turistas digitales son las siguientes:

- Potenciar las emociones.
- El eje principal son las redes sociales.
- El Smartphone como base para todo.
- Explorar las nuevas tecnologías.
- Reputación Online.
- Personalización a través del big data.

### 2.1. Soluciones actuales

Actualmente ya existen en el mercado diferentes aplicaciones o portales, tanto web como móvil, enfocadas al turismo. Algunos ejemplos de los proyectos anteriormente mencionados son los siguientes:

- **Tourist eye**

Aplicación móvil disponible para IOS y Android. Su principal utilidad es la de planificar viajes, se pueden realizar desde su propia web o desde la aplicación.

Cuenta con la posibilidad de que su comunidad agregue puntos de interés geolocalizados con su respectiva información. Estos puntos de interés pueden ser restaurantes, hoteles o cualquier sitio turístico. Podemos marcar los sitios como visitados y escribir notas.

Se pueden añadir ciudades a un itinerario y seleccionar las fechas en las que se va a viajar. Otra opción interesante es que se pueden descargar los mapas offline por lo que reducimos el consumo de datos móviles cuando estamos viajando.

Permite, además, que los usuarios creen rutas para otras personas. Cuenta con un historial de viajes y de sitios visitados que se guardan en el perfil del usuario.

Esta aplicación dejó de funcionar el 1 de diciembre de 2016 por lo que no se puede conseguir mucha más información. En su lugar apareció otra que comentaremos a continuación.

- **Guides by Lonely Planet**

Esta aplicación nació como heredera de tourist eye. Con más de 500.000 descargas es una de las aplicaciones más utilizadas. Cuenta con una serie de guías para descargar offline de diferentes ciudades en las que incluye toda la información relacionada con lugares de interés, como restaurantes, museos o clubs. De esta manera los usuarios pueden consultar la información sin necesidad de estar conectado a la red.

Ha perdido muchas funcionalidades con respecto a su antecesora ya que esta no cuenta con la opción de que los usuarios añadan datos. Su información es estática. Además, sólo está disponible en inglés, lo que podría suponer un problema para ciertas personas.

Otro de los problemas es que cuenta con un número limitado de guías. Es verdad que en su base de datos tenemos más de 200 ciudades, pero no nos permite visualizar nada más allá de estas.

Android:

<https://play.google.com/store/apps/details?id=com.lonelyplanet.guides&hl=es> 419

iOS:

<https://itunes.apple.com/es/app/guides-by-lonely-planet/id1045791869?mt=8>

- **Tripadvisor**

Sitio web por excelencia para la consulta de opiniones, comparar y reservar en restaurantes y hoteles. También cuenta con pequeñas guías de las ciudades con información sobre qué hacer, comentarios y foros donde los usuarios pueden realizar y responder preguntas.

Cuenta con un mapa que nos marca todos los lugares de interés, tanto turísticos como gastronómicos. Al entrar en alguno de estos lugares nos muestra una serie de imágenes subidas por la comunidad y una lista de opiniones de los mismos junto a su localización.

Enlace: <https://www.tripadvisor.es/>

- **Moovit**

Aplicación para saber “como llegar a un sitio”. Moovit es una aplicación, disponible en Android, IOS y con sitio web, que nos permite movernos con total libertad por una ciudad. Cuenta con una interfaz intuitiva y nos traza las diferentes rutas de cómo llegar de un sitio a otro, incluyendo transporte público con horarios en tiempo real. Actualmente cuenta con más de 2200 ciudades de 80 países diferentes. Recibe ayuda de la comunidad, que puede mapear rutas para otros usuarios e introducir información sobre el transporte público de ciudades alrededor del mundo.

Sitio web: <https://moovit.com/>

Android: <https://play.google.com/store/apps/details?id=com.tranzmate&hl=es>

iOS: <https://itunes.apple.com/es/app/moovit-transporte-p%C3%ABlico/id498477945?mt=8>

Como se puede observar hay una gran variedad de aplicaciones que nos sirven para solucionar diferentes tipos de problemas, ya sea en temas de transporte, guías de turismo etc. Pero todas carecen de algo, ninguna de ellas interactúa de manera extensa con los usuarios. Algunas como TripAdvisor cuentan con comentarios y puntuaciones de la comunidad. Pero ninguna nos permite añadir, por ejemplo, una localización que nos resulta interesante y no aparece en las guías o en los mapas. Este es uno de los problemas que aborda el trabajo que se está llevando a cabo, cuya principal diferencia frente al resto es una interacción completa con la comunidad para así proporcionar información que ninguna otra solución ya existente aporta.

## 2.2. Tecnologías para el desarrollo

Las tecnologías necesarias para realizar este trabajo se dividen en dos grandes grupos. Por una parte, tenemos las destinadas a la parte del servidor (backend), y por otra, las que se utilizan en la parte del cliente (frontend).

Para empezar a hablar sobre las tecnologías de desarrollo primero debemos saber a qué nos referimos cuando hablamos de “backend”.

El **backend** o CMS (Content Management System), en castellano, Sistema de Gestión de Contenidos, es la parte de una aplicación que el usuario final, es decir, los clientes, no pueden ver. Dentro de este grupo existen labores como accesos a las bases de datos o generación de plantillas del lado del servidor.

Las funciones principales del backend son las siguientes:

- Acceder a la información que se solicita a través de la aplicación cliente.
- Combinar la información encontrada y transformarla para que sea útil.
- Devolver la información.

Con la definición de backend ya hecha, vamos a definir a qué nos referimos cuando hablamos de “frontend”.

El **frontend** compone todas aquellas tecnologías que corren del lado del cliente. Una persona que trabaja sobre frontend es la encargada de maquetar la estructura del contenido, su diseño y agregar la interacción con el usuario. Estas 3 tareas se suelen realizar con 3 lenguajes; HTML, CSS y JavaScript.

Una vez clara la diferencia entre backend y frontend vamos a pasar a hablar del modelo de desarrollo software que se va a utilizar.

**La programación por capas** tiene como objetivo separar las diferentes partes que componen un sistema. La ventaja principal de este tipo de modelo es que, al estar el desarrollo dividido en distintos niveles o capas, si se realiza algún cambio, sólo afecta a la capa correspondiente, no es necesario modificar ni tocar el resto de capas. Estas capas son las siguientes:

- Capa de presentación: esta capa es la interfaz gráfica, donde se presenta el sistema al usuario, le comunica la información y captura los datos que pudiera introducir.
- Capa de negocio: aquí residen los programas que se ejecutan, reciben las peticiones de los usuarios y envían las respuestas correspondientes.

- Capa de datos: es la capa en la que residen los datos y la encargada de acceder a ellos. Aquí se encuentran los gestores de bases de datos.

### 2.2.1. Capa de datos

Una vez clara la definición de la capa de datos debemos pasar a hablar sobre el pilar de toda aplicación, la **base de datos**:

*“Una Base de Datos es un conjunto exhaustivo, no redundante de datos estructurados, organizados independientemente de su utilización y su implementación en máquina, accesibles a tiempo real y compatibles por usuarios concurrentes que tienen necesidad de información diferente y no predecible en el tiempo”.* Flory,1982.

Existen muchas maneras de diferenciar los tipos de bases de datos que existen, pero por defecto están divididas en dos grandes grupos, **Relacionales y No Relacionales**.

Las bases de datos **relacionales** son el modelo estándar y el más utilizado. Se manipulan mediante un lenguaje de peticiones robusto pero muy poco flexible. Siguen el modelo relacional, que basa su teoría en el uso de relaciones, esto quiere decir que se ve cada relación como si fuese una tabla formada por registros (tuplas) y columnas (campos). Las características de las bases de datos relacionales son las siguientes:

- Están compuestas de varias tablas o relaciones.
- No pueden existir dos tablas con el mismo nombre.
- Cada tabla es un conjunto de columnas y filas.
- La relación entre tablas se realiza mediante el uso de claves primarias y ajenas.
- Las CP (claves primarias) deben cumplir con la integridad de datos. Las CA (claves ajenas) se colocan en la tabla hijo, y coinciden con el valor de la CP del padre.
- Utilizan SQL (Structured Query Language) o lenguaje de consulta estructurada como interfaz principal para comunicarse.

Por otra parte, tenemos las bases de datos **no relacionales**, optimizadas para determinados modelos de datos que no tienen esquema y que tienden a escalar. Estas bases de datos difieren básicamente de las anteriores en que no utilizan SQL como lenguaje principal para realizar sus consultas. Suelen utilizarse para aplicaciones con un gran volumen de datos que requieren baja latencia, ya que no permiten extraer información relacionada entre tablas o lo hacen muy limitadamente.

	<i>Relacionales</i>	<i>No Relacionales</i>
<i>Ventajas</i>	<ul style="list-style-type: none"> <li>- Estándar SQL.</li> <li>- Mayor soporte</li> <li>- Permite el uso de transacciones en las operaciones.</li> </ul>	<ul style="list-style-type: none"> <li>- Abierta y flexible.</li> <li>- Alta escalabilidad.</li> <li>- Mínima necesidad de recursos para ejecutarse.</li> </ul>
<i>Desventajas</i>	<ul style="list-style-type: none"> <li>- No son flexibles.</li> <li>- Escalabilidad reducida.</li> </ul>	<ul style="list-style-type: none"> <li>- No estandarización</li> <li>- Pocas herramientas de monitoreo.</li> <li>- Poco compatibles con datos SQL</li> </ul>

*Tabla 1: Comparación de Bases de Datos*

Tanto si usamos bases de datos relacionales como no relacionales necesitamos un **sistema de gestión** para las mismas. Estos sistemas se encargan de almacenar, modificar y extraer información de una base de datos además de permitirnos realizar ciertas tareas como añadir, modificar o borrar los datos. Algunos de los sistemas existentes son:

**Oracle Database** [20]: Sistema de gestión de base de datos del tipo objeto-relacional. Desarrollada por Oracle. Esta arquitectura es muy utilizada por empresas ya que es muy escalable y se ejecuta en casi todas las plataformas existentes. Uno de sus puntos a favor es que su arquitectura se divide en dos sectores, la lógica y la física, esto permite que exista una mayor flexibilidad y robustez en lo que a los datos se refiere.

**Microsoft SQL Server** [13]: Sistema de gestión de bases de datos relacionales. Creado y distribuido por Microsoft, comparte muchas características con Oracle. Uno de sus fuertes es que se ejecuta en Transact-SQL, un conjunto de programas que nos permiten añadir diferentes características como la posibilidad de utilizar diferentes tipos de lenguaje de programación, un tratamiento de errores y excepciones, procesamiento de datos etc.

**MySQL** [16]: Otro gestor de base de datos relacional que pertenece a Oracle. Este en concreto es de código abierto y cuenta con licencia comercial. Es multiusuario y el más utilizado dentro de los de software libre. Algunos de sus puntos fuertes, aparte de ser gratuito, es que requiere poco procesador y memoria para funcionar, sin perder además velocidad en sus operaciones.

**MongoDB** [15]: Sistema de base de datos NoSQL, de código abierto y orientado a documentos. MongoDB guarda estructuras de datos con un esquema dinámico en formato similar a JSON (BSON) haciendo que la integración sea más rápida y fácil.

**Redis** [21]: Base de datos del tipo clave-valor apoyada por VMWare. Opcionalmente también puede ser utilizada como una base de datos durable o persistente. Es un software de código abierto escrito en ANSI C. Esta base de datos no permite realizar consultas, sólo puede obtener e insertar datos. Funciona en todas las plataformas, pero no tiene soporte en Windows.

### 2.2.2. Capa de negocio

Una vez investigado todo lo relacionado con las bases de datos, es momento de pasar a hablar sobre la siguiente capa. ¿Cómo construir aplicaciones y unir al cliente con el servidor?

Para realizar esta tarea contamos con varios frameworks y aplicaciones en el mercado que nos simplifican el trabajo.

**Node.js** [17]: Si nos basamos en la definición de su página web oficial:

“Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente.”

En otras palabras, Node.js utiliza JavaScript en el lado del servidor, lo que nos ofrece un diseño orientado a eventos y asincronía, gracias a esto, tenemos la capacidad de soportar un gran número de conexiones simultáneas en nuestro servidor de manera muy eficiente. Otro de los puntos a tener en cuenta es su gestor de paquetes NPM [18] (Node Package Manager). Son librerías Open Source desarrolladas por la comunidad.

Con Node.js podemos crear un servidor web, realizar peticiones a una base de datos y devolver los resultados mediante una API, subir archivos a un servidor...

**VertX** [23]: Framework de Java que se ejecuta en la JVM (Java Virtual Machine) y nos permite construir aplicaciones. Su programación es parecida a Node.js ya que utiliza el lenguaje JavaScript en el servidor, pero trabaja de forma reactiva enfocándose en microservicios.

Los microservicios son elementos que funcionan de manera autónoma. Además de esto, su código debe ser desplegado de manera que no afecte a ninguno de los otros microservicios.



VertX es escalable gracias a su uso de microservicios y poliglota, lo que quiere decir que podemos utilizarlo, no sólo con JavaScript, sino con otros muchos lenguajes como pueden ser Java, Ruby, Ceylon...

**Spring:** Framework de Java para el desarrollo de aplicaciones con inversión de control e inyección de dependencia.

La inversión de control (la inyección de dependencia es un tipo de inversión de control) es, básicamente, el cambio de flujo de ejecución y vida de los objetos, esto quiere decir que cosas que antes dependían del programador, como puede ser el orden de llamada de métodos, ahora depende del framework. Esto nos permite realizar aplicaciones más complejas y con funcionamientos más automáticos sin necesidad de invertir más tiempo.

Spring se basa en ficheros XML para construir él solo los objetos que necesita nuestra aplicación.

Cuenta con una serie de módulos que nos permiten realizar diferentes acciones como acceder a datos y herramientas de mapeo de objetos relacionales en bases de datos NoSQL, procesos de seguridad configurables...

### 2.2.3. Capa de presentación

Cuando se trata de diseñar una aplicación que va a llegar a un gran número de personas distintas hay varios factores a tener en cuenta a la hora de elegir las tecnologías con las que vamos a trabajar, ya que no todas van a satisfacer nuestras necesidades de cara a tener un producto funcional o completamente operativo.

Como ya sabemos, este proyecto consiste en realizar una aplicación para dispositivos móviles. Lo primero con lo que nos encontramos es si debemos crear una aplicación nativa, híbrida o generada. Para poder elegir, lo primero es entender cómo funciona cada una y realizar una comparación.

**Nativas:** Las aplicaciones nativas son las que se desarrollan única y exclusivamente para dispositivos o, mejor dicho, sistemas operativos móviles. Esto significa que si queremos realizar una aplicación enfocada tanto para Android como para iOS tenemos que hacer dos proyectos diferentes, ya que cada uno de ellos tiene lenguajes de programación diferentes. Android hace uso de Java mientras que iOS utiliza Objective-C o Swift. Por lo general, este tipo de aplicaciones tienen un mayor rendimiento y aprovechan mejor el hardware del dispositivo (GPS, cámara).

**Híbridas:** Estas aplicaciones se desarrollan utilizando tecnologías web (HTML, CSS, JavaScript) y son procesadas después a través de Frameworks. Estas aplicaciones móviles sólo necesitan programarse una vez, y funcionan para varios sistemas operativos, es lo que se conoce como “Aplicación multiplataforma”. Cabe destacar que estas aplicaciones no pueden acceder al hardware del dispositivo de manera directa, es necesario el uso de plugins para esto.

**Generadas:** Este tipo de aplicaciones son aquellas que utilizan un lenguaje de programación específico para luego generar automáticamente el código nativo para cada plataforma. Por ejemplo, si utilizamos Xamarin, escribiríamos el código de la aplicación en C# para que después el compilador genere el código nativo que sea requerido. Esta tecnología aún está en desarrollo por lo que cuenta con algunos problemas.

La mejor manera de ver las diferencias y ventajas de unas aplicaciones frente a otras es la realización de una tabla comparativa más visual.

	<i>Nativas</i>	<i>Híbridas</i>	<i>Generadas</i>
<i>Multiplataforma</i>	NO	SI	SI
<i>Rendimiento</i>	ALTO	MEDIO	MEDIO
<i>Acceso Hardware</i>	SI	PLUGINS	NO
<i>Costo de desarrollo</i>	ALTO	BAJO	MEDIO

*Tabla 2: Comparación tipos de Aplicaciones*

En el mercado existen diversos frameworks para la realización de aplicaciones. Dependiendo de nuestros conocimientos o necesidades nos decantaremos por uno u otro.

**PhoneGap [1]:** Es el framework multiplataforma más conocido a la hora de realizar aplicaciones híbridas. Producido por Nitobi y comprado posteriormente por Adobe Systems. Es una distribución de código abierto de Cordova que utiliza para renderizar CSS3 y HTML5 y JavaScript para la lógica. Gracias a HTML5 puede acceder a diferentes partes del hardware (GPS, acelerómetro, cámara) del dispositivo sin necesidad de utilizar

plugins o programas de terceros, aunque cuenta con un gran número de estos para ampliar la funcionalidad si así se desea. Sin embargo, tiene algunos problemas ya que no todos los navegadores móviles son compatibles con HTML5.

**Framework 7 [5]:** Framework de HTML móvil de fuente abierta y gratuita para el desarrollo de aplicaciones móviles híbridas o aplicaciones web. Basa su funcionamiento en la creación de aplicaciones para iOS ya que su diseño se enfoca en este tipo de dispositivos. Framework 7 no es compatible con todas las plataformas (se centra únicamente en iOS como hemos dicho) pero con algunos programas de terceros podemos hacer que sea compatible con dispositivos Android e incluso con un navegador web. Cuenta con su propia librería JS y puede ser integrado con PhoneGap o Cordova, pero no lo soporta de manera oficial.

Por último, también nos permite desarrollar de manera sencilla prototipos para aplicaciones, ya que no necesita mucho código para desempeñar esta tarea.

**Ionic [9]:** Es el framework más completo y popular para la creación de aplicaciones híbridas. SDK (Software Development Kit) de código abierto que funciona sobre AngularJS [2] y Apache Cordova. Gracias a AngularJS consigue páginas más dinámicas y estructuración por componentes además del modelo vista-controlador (MVC) que proporciona una separación entre la lógica de la aplicación y su apartado visual.

Su desarrollo está basado en componentes, lo que quiere decir que serán más escalables y sostenibles. Gracias a esto podremos solucionar pequeños problemas sin ninguna dificultad. Cuenta con una gran comunidad muy activa.

Como añadido cuenta con muchas características nativas de dispositivos (autenticación de huellas dactilares, Bluetooth...)

**Xamarin [14]:** Empresa que más ha crecido en los últimos años de la mano de Microsoft (quien compró a la compañía). Xamarin es un entorno de desarrollo de aplicaciones móviles que utiliza C# (lógica) y XAML (vistas), de esta manera consigue compilar de manera nativa las aplicaciones para múltiples plataformas, es decir, es un tipo de aplicación generada. Este entorno nos permite realizar un único código para luego transformarlo y ser utilizado en la plataforma que nosotros queramos.

**React Native [4]:** Mobile framework JavaScript, propiedad de Facebook, que utiliza los componentes de ReactJS (Librería JavaScript) para crear aplicaciones móviles. Cuenta con un intérprete que lee los HTML y coloca los componentes nativos en su posición.

Esto significa que todas las interacciones (frontend) con el usuario se realizarán de forma nativa.

### 3. Objetivos

A la hora de empezar a realizar un trabajo como este, debemos tener claros cuáles son los objetivos o metas que se van a cubrir. Para ello dividimos los objetivos en dos categorías: un objetivo general y los objetivos específicos.

El **Objetivo general** es desarrollar una aplicación móvil de índole turístico que permita a los usuarios visualizar los lugares de mayor interés en la zona en la que se encuentran gracias a la ayuda de un mapa con información personalizada y poder dar de alta otros lugares para que futuros usuarios los puedan disfrutar. Las características principales con las que debe contar la aplicación son:

- Aplicación multiplataforma.
- Diseño llamativo e intuitivo para todo tipo de usuarios que proporcione una buena experiencia de usuario.
- Acceso a la ubicación en tiempo real.

Los objetivos específicos son los siguientes:

- **Objetivo específico 1:** Establecer cuáles van a ser los requisitos y especificaciones de la aplicación.
- **Objetivo específico 2:** Conocer las tecnologías que existen para llevar a cabo la aplicación.
- **Objetivo específico 3:** Desarrollar una base de datos con el sistema de gestión MySQL.
- **Objetivo específico 4:** Utilizar la herramienta Node.js para crear un backend ligero y eficiente.
- **Objetivo específico 5:** Unir backend y frontend creando un servicio API REST que, mediante peticiones HTTP permitan al cliente comunicarse con el servidor.
- **Objetivo específico 6:** Utilizar la herramienta Ionic3 para poder realizar una aplicación lo más eficiente y usable posible.
- **Objetivo específico 7:** Conocer los límites y posibilidades que nos ofrece la API de Google Maps.
- **Objetivo específico 8:** Crear un foro dinámico de preguntas y respuestas para los usuarios.

## 4. Metodología

Para la realización de este trabajo se estimó una duración aproximada de 5 meses en los cuales el tiempo se dividiría de la siguiente manera:

- El primer mes fue destinado a la investigación y búsqueda de información relacionada con los diferentes lenguajes y maneras de llevar a cabo el proyecto, además de empezar a crear bocetos muy básicos sobre cómo quedaría la aplicación.
- El segundo, tercer y cuarto mes fueron destinados a la realización de las diferentes partes de la aplicación, en este caso, el backend (toda la configuración del servidor, su seguridad y la API) y el frontend (realización de todas las funcionalidades y parte de la maquetación).
- El quinto y último mes fue reservado para terminar de maquetar los pequeños puntos de la aplicación, añadir algunas de las funcionalidades más prescindibles (dependiendo del tiempo restante) y la realización de las pruebas de la aplicación para comprobar que todo funciona a la perfección.

Con respecto a la memoria del trabajo, se fue adelantando conforme se iba avanzando en la aplicación. Los primeros meses se cubrirían todos los puntos relacionados con la investigación, los bocetos, introducción etc. Y el resto de meses, conforme se iba avanzando en el desarrollo y diseño de la aplicación, se iba rellenando sus puntos correspondientes.

Para llevar un control de lo que se tiene y de lo que se ha desarrollado se ha decidido utilizar una metodología de desarrollo ágil (en este caso SCRUM). La razón de utilizar esta metodología es que ya ha sido utilizada con anterioridad por el alumno con buenos resultados.

Para seguir esta metodología se ha utilizado Trello. Trello es una herramienta que nos permite, mediante tarjetas y listas, gestionar todo el proceso de nuestro trabajo, desde añadir las tareas a realizar, hasta marcarlas como realizadas o en proceso.

En concreto el “tablero” que se ha utilizado para gestionar el trabajo cuenta con 4 listas (ver Figura 1).

- TO DO
- DOING
- DONE (APP)
- DONE (Documento)

Se ha optado por dividir las tareas ya realizadas en 2 listas para así distribuir mejor la información.

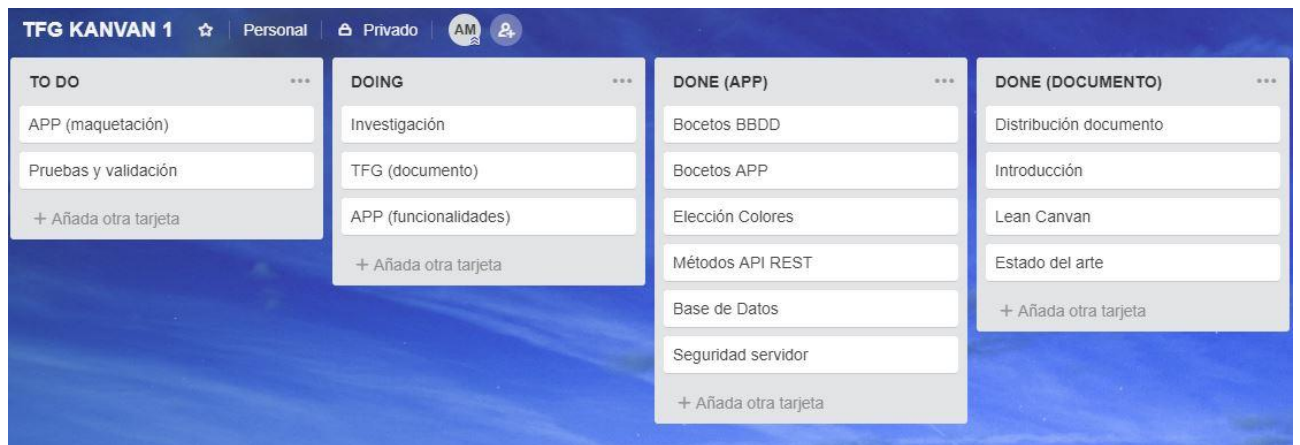


Figura 1: SCRUM

## 5. Análisis y especificación

A la hora de empezar a realizar un proyecto o producto, el primer paso a seguir para poder desarrollarlo es definirlo de forma correcta.

Para ello, primero hay que analizarlo, esto permite al usuario que lo desarrolla especificar las características con las que va a contar el producto, sus restricciones etc.

Una vez hecho esto se debe especificar los requerimientos del producto. Esto consiste en una descripción completa del sistema que se va a desarrollar, con todas sus interacciones y funcionalidades. Estos requisitos se dividen en dos grupos:

- Requisitos funcionales: Describen el conjunto de entradas, comportamientos y salidas que el sistema va a proporcionar. Cada uno de ellos cuenta con una descripción de los pasos a seguir para llevar a cabo una funcionalidad.
- Requisitos no funcionales: requisitos que definen las características en el comportamiento del producto. Son utilizados para imponer las diferentes restricciones, tanto de diseño como de funcionalidad o disponibilidad.

### 5.1. Lienzo de modelo de negocio (Lean Canvas)

El Lean Canvas es una herramienta de visualización de modelos de negocio que une elementos de dos modelos diferentes: Business Model Canvas y Lean Startup.

El Business Model Canvas es utilizado por empresas que ya poseen una dinámica de trabajo en funcionamiento. El problema de este método es que cuentan con diversos bloques que no existen dentro de la fase “startup” de un proyecto o empresa.

Por todo esto, Ash Maurya decidió unir los dos modelos para crear uno que fuese adecuado para empresas o proyectos que están empezando.

El contenido de los bloques adaptado a este proyecto (ver Figura 2: Lean Canvas) son los siguientes:

#### **Problema**

A nivel personal, algunos de los problemas que se encuentran las personas cuando viajan son, principalmente, la falta de información sobre el lugar que visitan, los lugares que pueden resultar de su agrado o la manera de llegar a ellos.



Otro de los problemas es que no saben dónde o a quién preguntar si tienen dudas sobre su próximo destino.

Además de los dos anteriores, que son más generales, otro problema a tener en cuenta es la necesidad de algunas personas de compartir sus experiencias o querer mostrarle al resto de usuarios los lugares que han encontrado durante su viaje y que consideran de interés.

### **Solución**

La solución que esta aplicación propone a los problemas anteriores es la integración de un mapa que nos muestre, tanto los puntos de interés de un lugar o zona, como los que haya creado la comunidad (con toda la información y fotos correspondientes), además de la posibilidad de trazar y visualizar una ruta para llegar hasta ellos.

Otra de las soluciones es la creación de un foro para que la comunidad pueda añadir preguntas y responder otras.

### **Segmento de clientes**

Contamos con 3 tipos de clientes potenciales que utilizarían la aplicación. En primer lugar, tenemos a los turistas, tanto extranjeros como nacionales, que cuenten con un dispositivo móvil. El siguiente tipo de cliente es el que reside o visita un lugar, y quiere compartir con el resto lugares de interés que, por norma general, no aparecen en las guías. El último segmento de clientes es el que necesita ayuda a la hora de viajar y no sabe dónde o a quién preguntar ciertas dudas.

### **Proposición de valor única**

La aplicación móvil contará con una extensa cantidad de información a disposición de los usuarios sobre su próximo destino turístico, con lugares personalizados, fotos y comentarios sobre estos lugares, además de un foro donde poder consultar cualquier duda que les pueda surgir antes o durante su viaje.

### **Ventaja especial**

La principal ventaja que nos proporciona esta aplicación es la extensa lista de lugares personalizados que contendrá, y que ninguna otra aplicación tiene, ya que son creados por los mismos usuarios en la plataforma. Otra de las ventajas es que cuenta con un foro, dividido por ciudades, a disposición de los usuarios.

### **Canales**

Los canales de distribución de la aplicación son principalmente tres:

- Play Store (Tienda oficial de dispositivos con el sistema operativo Android).
- AppStore (Tienda de aplicaciones de dispositivos Apple).
- Foros de viajes.

### **Estructura de costes**

Los gastos principales que supondrán la creación y despliegue de la aplicación son los siguientes:

- Mano de obra (Personal que trabaje en la creación y mantenimiento de la aplicación).
- Servidor Web.
- Mantenimiento del servidor.

### **Flujo de ingresos**

Existen dos maneras de obtener ingresos con la aplicación.

La primera manera consiste en el número de descargas con las que cuente la aplicación, ya que, si son muy elevadas, esto nos generará más ingresos y más visibilidad en las diferentes tiendas móviles.

La otra manera es la inclusión en la aplicación de anuncios de terceros que quieran obtener visibilidad.

### **Métricas clave**

Las métricas a tener en cuenta en esta aplicación son tanto el número de descargas que tenga la aplicación como las veces que los usuarios vuelvan a utilizarla. Además de esto también será importante el grado de interacción del usuario con la plataforma (creación de nuevos lugares, foro activo...).

<b>Problema</b>  - Las personas no conocen todos los lugares de interés o no saben como llegar a ellos.  - No pueden compartir con otros lugares desconocidos.  - No saben a quien o donde preguntar ciertas dudas sobre su viaje.	<b>Solución</b>  - Mapa con destinos de interés tanto generales como añadidos por la comunidad.  - Información detallada de estos lugares.  - Foro para resolver dudas.  <b>Métricas Clave</b>  - Descargas de la app.  - Usuarios recurrentes.  - Interacción con la app (foro, nuevos lugares).	<b>Proposición de valor única</b>  Aplicación móvil con una extensa información sobre su destino turístico y foro para realizar y responder preguntas con otros usuarios.	<b>Ventaja especial</b>  Lugares de interés que no tienen otras aplicaciones o buscadores ya que los crean la comunidad, foro de ayuda a otros viajeros.  <b>Canales</b>  - Play Store.  - AppStore.  - Foros de viajes.	<b>Segmentos de clientes</b>  - Turistas nacionales.  - Turistas extranjeros.  - Personas que quieren compartir sus experiencias viajando.  - Personas que quieren ayudar a otros cuando viajan.
<b>Estructura de costes</b>  - Mano de obra.  - Servidor.  - Mantenimiento del servidor.			<b>Flujos de ingresos</b>  - Descargas de la aplicación.  - Publicidad de terceros en la app.	

Figura 2: Lean Canvas

## 5.2. Requisitos funcionales

Código	RF1
Requisito	Login
Prioridad	Alta
Descripción	El usuario debe ser capaz de registrarse en nuestra aplicación para, posteriormente, utilizar su Nick y contraseña para poder iniciar sesión y así poder utilizar todas las funcionalidades de la aplicación. Una vez iniciada sesión, esta información se guarda durante todo el rato que utilicemos la aplicación.

Código	RF2
Requisito	Creación lugares personalizados
Prioridad	Alta
Descripción	El usuario tendrá la posibilidad de poder rellenar un formulario con información de un lugar de interés para que otros usuarios puedan hacer uso de esta información.

Código	RF3
Requisito	Ver Mapa
Prioridad	Alta
Descripción	El usuario podrá hacer uso de un mapa generado gracias a la API de Google Maps. En este mapa podrá ver los puntos de interés guardados en nuestra aplicación o los que ya existan en la API de Google.

Código	RF4
Requisito	Valorar lugar
Prioridad	Alta
Descripción	El usuario, una vez logueado y registrado, podrá puntuar un lugar personalizado creado por otro usuario para así ayudar a la comunidad y darle prioridad a unos lugares u otros.

Código	RF5
Requisito	Comentar Lugar
Prioridad	Alta
Descripción	El usuario podrá añadir un comentario de un lugar personalizado para dar información adicional a la comunidad o simplemente, opinar sobre él y sus alrededores.

Código	RF6
Requisito	Ver Foro
Prioridad	Alta
Descripción	El usuario tendrá a su disposición un pequeño foro con preguntas y respuestas de la comunidad sobre turismo en diferentes ciudades.

Código	RF7
Requisito	Realizar Pregunta
Prioridad	Alta
Descripción	El usuario, una vez registrado y logueado, podrá lanzar una pregunta en el foro para que otros usuarios puedan visualizarla.

Código	RF8
Requisito	Responder pregunta
Prioridad	Alta

Descripción	El usuario, una vez registrado y logueado, podrá acceder a la lista de preguntas de la comunidad y responder a ellas para resolver las dudas de otros usuarios.
-------------	---

Código	RF9
Requisito	Cámara y galería
Prioridad	Alta
Descripción	La aplicación podrá acceder tanto a la cámara del dispositivo como a la galería para poder subir al servidor imágenes de los lugares que demos de alta.

Código	RF10
Requisito	Cerrar Sesión
Prioridad	Media
Descripción	El usuario tendrá la posibilidad de cerrar sesión en el dispositivo en el que está utilizando la aplicación.

Código	RF11
Requisito	GPS
Prioridad	Media
Descripción	La aplicación pedirá permiso al usuario para utilizar el GPS del dispositivo para así mostrarnos la información cercana a nuestra posición actual.

Código	RF12
Requisito	Editar Lugar
Prioridad	Media
Descripción	El usuario tendrá a su disposición la posibilidad de modificar un lugar personalizado que haya dado de alta él mismo.

### 5.3. Requisitos no funcionales

#### Disponibilidad

El sistema podrá ser utilizado las 24 horas del día ya que la API de Google Maps, salvo fallo interno, funciona siempre y la API propia de la aplicación está alojada en un servidor que está activo todo el tiempo.

## **Seguridad**

Todas las peticiones a la API propia de la aplicación se hacen mediante la utilización de JSON Web Tokens [11] para verificar la seguridad en las llamadas y, sobre todo, para mantener la sesión del usuario activa. Gracias a los JWT hay ciertas peticiones a la API que no podrán ser realizadas sin tener un Token de acceso. Además de esto, las contraseñas de los usuarios que se registran en nuestra plataforma se guardan encriptadas, de manera que, aun obteniendo los datos de la base de datos de la aplicación, no podrían acceder a sus datos. Por último, mencionar que los accesos a la base de datos desde la API no se realizan desde el código base de los métodos, sino que cuenta con un archivo aparte donde se guardan los datos de acceso y el código secreto de los Tokens.

## **Rendimiento**

Todas las peticiones y las diferentes páginas de nuestra aplicación deben ser capaces de responder en tiempo real y con una velocidad de carga alta. Esto lo conseguimos gracias a la eficacia de Node.js, la optimización del código en el frontend y de la compresión de fotos que se realiza al subir una imagen al servidor.

## **Multiplataforma**

La aplicación debe ser completamente accesible y operativa en todos los sistemas operativos móviles que se utilizan en este momento (Android, iOS, Windows phone). Esto es posible gracias a la utilización del framework Ionic3.

## **5.4. Tecnologías Utilizadas**

Con toda la información recogida hasta el momento de las funcionalidades con las que queremos que cuente la plataforma, sus requisitos y el alcance que queremos que tenga, podemos hablar de las tecnologías que nos ayudarán a llevar a cabo esta tarea.

### **MySQL**

MySQL es el sistema de gestión de base de datos que contiene este proyecto. En él está alojada la base de datos que envía la información a la aplicación. Las razones por las que se ha optado por este sistema son, principalmente, que utiliza el estándar SQL, es de código abierto y proporciona un muy buen rendimiento.

La versión de MySQL utilizada es la 5.7.20.

## MySQL Workbench

Workbench es una herramienta visual de bases de datos. Con ella importamos y exportamos las bases de datos, si necesitamos una copia de la misma.

Esta herramienta ha sido utilizada para crear desde cero toda la estructura de la base de datos una vez terminado el esquema final (Figura 5: Esquema de la base de datos) además de otras opciones como son la inserción de datos de ejemplo o auxiliares y para comprobar que desde la propia aplicación se realizan bien todas las consultas, ya sea ver que devuelve la información pedida o que añade y elimina de forma correcta.

La versión de MySQL Workbench utilizada es la 6.3

## NodeJS

NodeJS es el programa encargado de gestionar todas las acciones que se realizan por parte del servidor. Este programa nos permite de manera muy sencilla crear un pequeño servidor que conecta con la base de datos y gestiona las peticiones entre los dos lados (la base de datos y el cliente).

Todas estas acciones las realiza gracias a un paquete llamado **Express** [3] que es una infraestructura de aplicaciones web, lo que nos permite contar con un gran número de características para aplicaciones móviles. En concreto este proyecto utiliza el paquete express porque permite crear de manera simple e intuitiva una API REST.

En resumen, NodeJS proporciona a este proyecto la herramienta sobre la que va a ejecutarse la API RESTFUL del proyecto necesaria para que el cliente obtenga y envíe los datos correspondientes, ya que sirve de conexión entre la base de datos y la aplicación. Además de todo esto también es donde se alojan todos los sistemas de seguridad (protección ante inyección de código, autenticación basada en token, encriptación de contraseñas).

La versión de Node utilizada es la 6.11.5.

## Ionic

Ionic es el framework utilizado para desplegar y diseñar todo el lado del cliente. Al ser una herramienta que genera aplicaciones híbridas, no es necesario crear una aplicación diferente para cada tipo de sistema operativo móvil. Otra de sus ventajas es que trabaja sobre Angular (en este caso la versión 4, que es más ligera y ocupa menos espacio), esta herramienta cuenta

con una gran comunidad de desarrolladores que, unido a su crecimiento escalable y su alto rendimiento lo hace una buena opción a la hora de ser utilizado en este proyecto.

Otro de los motivos por el cual se ha decidido utilizar Ionic es por su sencillez a la hora de integrar la API de Google Maps, ya que es un factor muy importante en el proyecto.

La versión de Ionic utilizada es la 3.19.



## 6. Diseño

En este apartado se define todo lo relacionado con la fase de diseño. Se establecerán las pautas o pasos a seguir para realizar la implementación de todo el sistema.

En los siguientes puntos se incluye la información relacionada con la arquitectura de todos los elementos que conforman la aplicación, además del diseño de las interfaces y los modelos de datos correspondientes a la base de datos.

### 6.1. Arquitectura de la aplicación

La arquitectura de esta aplicación está basada en capas, lo que quiere decir que cada capa es independiente de la otra, realiza sus propias funciones y se comunica con el resto para llevar a cabo una funcionalidad específica dentro del sistema REST.

La REST (representational state transfer o transferencia de estado representacional en castellano) es un tipo de arquitectura software para sistemas que basan su funcionamiento en la interacción de una interfaz que utilice el protocolo HTTP para obtener algún tipo de dato o para realizar una operación sobre él. Las características que definen a una arquitectura REST son las siguientes:

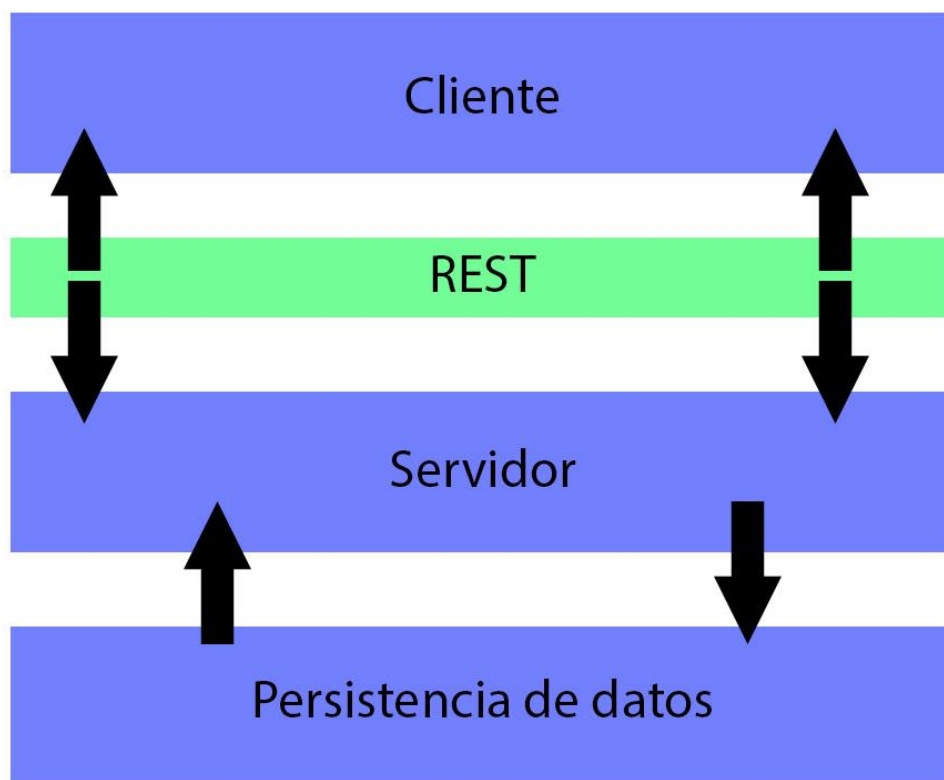
- Protocolo Cliente/Servidor sin estado: cada petición HTTP que se realiza contiene toda la información que es necesaria para llevarla a cabo. No se necesita ninguna información almacenada.
- Operaciones: Cuenta con diferentes operaciones, de las cuales utilizaremos GET (obtener), POST (crear), PUT (modificar), DELETE (eliminar).
- Sintaxis universal: Ya que realiza cada una de las acciones mediante su URI.
- Sistema de capas: Cuenta con una arquitectura jerárquica y cada capa realiza su función dentro del sistema.
- Interfaz uniforme, es decir, todo sigue un mismo esquema para que sea más fácil identificar las acciones.

El diseño de nuestra aplicación está formado por 3 capas:

- Capa de presentación: es la capa más externa del sistema, la que ve el usuario. Presenta el sistema al usuario, le muestra la información y obtiene la que el usuario

quiere mandar al sistema. Esta capa se comunica con la de negocio y está formada por el Cliente representada en la Figura 3.

- Capa de negocio: esta capa es la que recibe las peticiones del usuario y envía las respuestas tras procesar la petición. En esta capa es donde se encuentran todas las reglas o restricciones que deben cumplirse para que el sistema funcione. Se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados de estas solicitudes y con la capa de datos para guardar o recibir información. En esta capa también es donde se alojan los diferentes programas que se ejecutan. Esta capa está representada como “servidor” y REST en la Figura 3.
- Capa de datos: Es la capa en la que se alojan los datos y la que accede a ellos. La forma un gestor de bases de datos que es el que se encarga de recibir y enviar datos a la capa de negocio. En esta capa es donde se encuentra la base de datos de la aplicación. Está representada como “Persistencia de datos” en la Figura que se muestra a continuación.



*Figura 3: Arquitectura de la aplicación*

Una vez descritas las diferentes capas vamos a profundizar en la arquitectura de cada una de ellas:

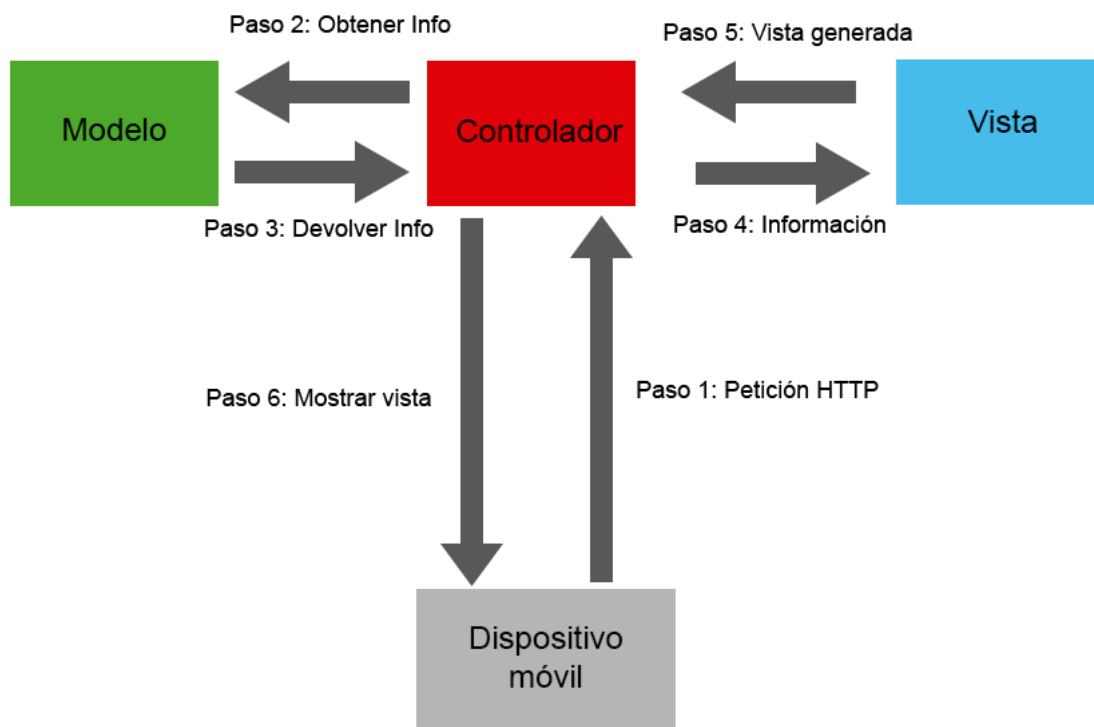
## **Arquitectura del servidor**

La arquitectura de nuestro servidor es bastante sencilla. Cuenta con un enlace entre la base de datos y el cliente. Cualquier cliente puede realizar peticiones a la API REST de la aplicación, pero solo obtendrá resultados si antes se autentica. Esto es posible gracias a que contamos con una seguridad basada en Tokens que nos permite “proteger” las rutas que creamos convenientes para que no sean accesibles a no ser que antes el cliente se haya identificado en la plataforma y hayamos recibido el token pertinente. De esta manera cada vez que se intente acceder a alguna ruta de la API REST se debe adjuntar el token correspondiente, de lo contrario, esta devolverá un mensaje de error.

## **Arquitectura del cliente**

Para desarrollar y diseñar el cliente se hará uso de Ionic3, que es un framework de Angular que permite la creación de aplicaciones móviles multiplataforma de manera muy eficiente. Ionic trabaja con el modelo MVC (Modelo-Vista-Controlador), o lo que es lo mismo, está basado en la idea de separar conceptos sin perder el diseño.

- **Modelo:** Es la capa más interna, la encargada de los datos, de realizar las peticiones a las bases de datos y de enviarles o recibir la información. Estas peticiones las realiza de manera local dentro de la propia aplicación o a una base de datos remota.
- **Vista:** Capa externa, la que ve el cliente. Aquí se genera el código que muestra los datos recibidos por el modelo para que los visualice el usuario.
- **Controlador:** Capa que enlaza las dos anteriores, se comunica con el modelo para actualizar su estado y con la vista para cambiar la información que muestra.



*Figura 4: Arquitectura Modelo Vista Controlador*

También cabe mencionar el uso de la última versión de ionic ya que cuenta con diferentes novedades respecto a las anteriores que optimizan el funcionamiento, como el uso de Angular 4.0 que hará a la aplicación menos pesada y más rápida.

## 6.2. Base de datos

La base de datos de la aplicación está formada por 5 tablas unidas entre ellas. Estas tablas son las siguientes:

- **Usuarios:** Tabla principal que contiene la información y los datos de los usuarios registrados en nuestra aplicación. Cabe destacar que las contraseñas se guardan de manera encriptada gracias a un paquete del que hablaremos más adelante. Los datos que recoge esta tabla son el Nick y la contraseña del usuario, además de su nombre y su ciudad de residencia.
  - Campos: Nick, Password, Nombre, Ciudad.
  - Clave Primaria: Nick.

- **Lugares:** Esta es la tabla que recoge la información de los lugares personalizados que crea la comunidad. Cuenta con un Identificador, una descripción, el tipo de lugar, su posición en el mapa y el Nick del usuario que lo ha creado.
  - Campos: ID, Título, Descripción, Tipo, Lat, Long, Ciudad, Votos, Puntuación, UsuarioL.
  - Clave Primaria: ID.
  - Clave Ajena: UsuarioL > Usuarios.
  
- **Comentarios:** Los usuarios podrán realizar comentarios sobre los lugares creados por la comunidad. Para esto contamos con una tabla con un Identificador y con el contenido del comentario en cuestión. También recoge la información del usuario que escribe el comentario y del lugar al que va dirigido.
  - Campos: ID, Contenido, Fecha, UsuarioC, LugarC.
  - Clave Primaria: ID.
  - Clave Ajena: UsuarioC > Usuarios, LugarC > Lugares.
  
- **Preguntas:** La aplicación también cuenta con un pequeño foro de preguntas y respuestas para la comunidad. Para la información de las preguntas tenemos un identificador, la pregunta y el usuario que la escribe.
  - Campos: ID, Contenido, Fecha, Ciudad, UsuarioP.
  - Clave Primaria: ID.
  - Clave Ajena: UsuarioP > Usuarios.
  
- **Respuestas:** Tabla que registra las respuestas de los usuarios a preguntas ya creadas. Cuenta con un Identificador, la respuesta, el usuario que la crea y el identificador a la pregunta que responde.
  - Campos: ID, Contenido, Fecha, UsuarioR, PreguntaR.
  - Clave Primaria: ID.
  - Clave Ajena: UsuarioR > Usuarios, PreguntaR > Preguntas.

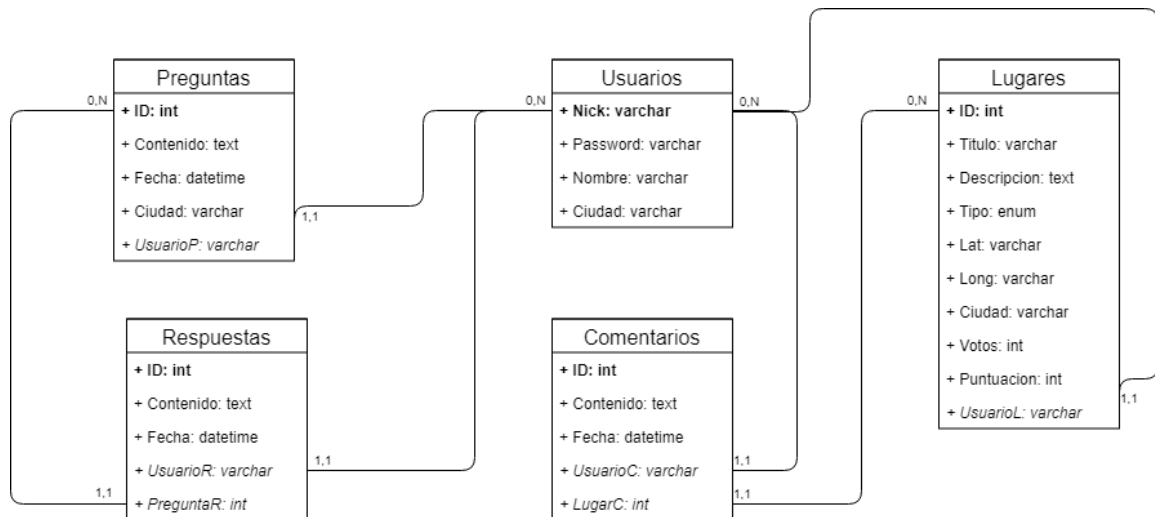


Figura 5: Esquema de la base de datos

### 6.3. API

Método HTTP	URI	Valores de entrada	Descripción
GET	/users/{nick}	-	Información del usuario con el Nick introducido.
POST	/users	Nick, password, nombre, ciudad	Registro de un usuario en la plataforma.
DELETE	/users/{Nick}	-	Eliminar un usuario de la plataforma.
POST	/sessions	Nick, password	Login del usuario.
GET	/users/{Nick}/places	-	Lista de los lugares dados de alta por el usuario.
GET	/users/{Nick}/comments	-	Lista de comentarios escritos por el usuario.
GET	/users/{Nick}/questions	-	Lista de preguntas hechas por el usuario.

GET	/places/cities/{ciudad}	-	Lista de los lugares creados por la comunidad de una ciudad en concreto.
POST	/places	Título, Descripción, Latitud, longitud, ciudad, usuario.	Crear un lugar de interés personalizado.
DELETE	/places/{idlugar}	-	Borrar lugar.
GET	/places	-	Información de todos los lugares creados.
GET	/places/{tipo}	-	Información de los lugares de un tipo en concreto.
GET	/places/{idlugar}	-	Información detallada de un lugar
POST	/places/{idlugar}/comments	Contenido, usuario	Añadir un comentario sobre un lugar
GET	/places/{idlugar}/comments	-	Comentarios de un lugar
POST	/places/{idlugar}/votes	puntuación	Añadir valoración a un lugar
GET	/questions/{idpregunta}		Mostrar la información de una pregunta
GET	/questions/cities	-	Lista ciudades con preguntas-
GET	/questions/cities/{ciudad}	-	Preguntas de una ciudad en concreto.
GET	/questions	-	Mostrar todas las preguntas
POST	/questions	Contenido, ciudad, usuario	Crear una pregunta en el foro.

DELETE	/questions/{idpregunta}	-	Borrar pregunta.
GET	/questions/{idpregunta}/answers	-	Ver respuestas de una pregunta.
POST	/questions/{idpregunta}/answers	Contenido, usuario	Crear una respuesta a una pregunta.
DELETE	/answers/{idrespuesta}	-	Borrar una respuesta.
POST	/places/{idlugar}/fotos/0	Imagen	Subir la imagen principal de un lugar.
GET	/places/{idlugar}/fotos/0	-	Ver la imagen principal de un lugar.
POST	/places/{idlugar}/fotos/1	Imagen	Subir una imagen secundaria de un lugar.
GET	/places/{idlugar}/fotos/1	-	Ver Imagen secundaria de un lugar.
POST	/places/{idlugar}/fotos/2	Imagen	Subir otra imagen de un lugar.
GET	/places/{idlugar}/fotos/2	-	Ver Imagen secundaria 2 de un lugar.
POST	/places/{idlugar}/fotos/3	Imagen	Subir tercera imagen de un lugar.
GET	/places/{idlugar}/fotos/3	-	Ver Imagen secundaria 3 de un lugar.

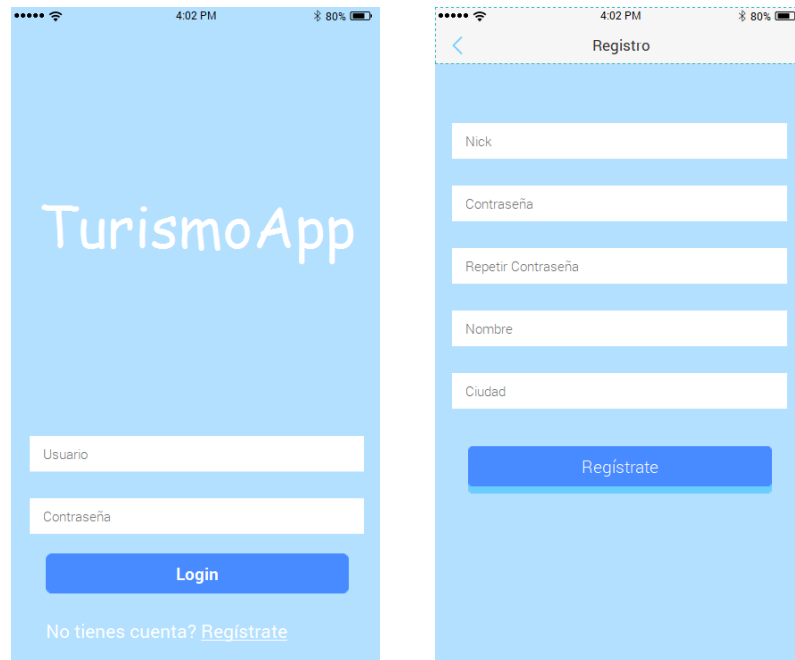
Tabla 3: API Rest

## 6.4. Interfaces

Antes de empezar a programar debemos tener claro cómo van a ser las vistas de nuestra aplicación y cómo va a ser la interacción entre ellas. Para ello hemos utilizado la herramienta Justinmind [12] que nos permite realizar bocetos bastante realistas de cómo sería una



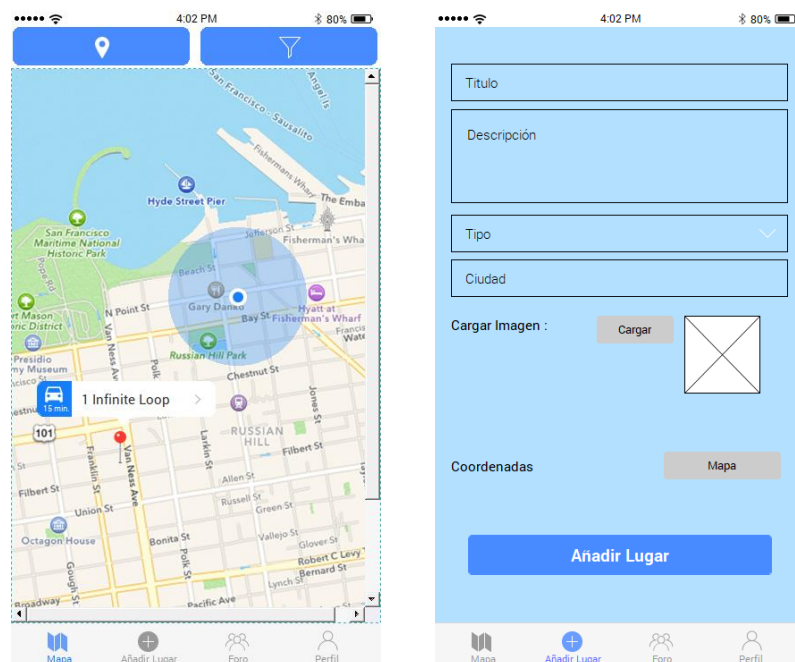
aplicación móvil y además unir las diferentes páginas mediante la interacción de botones o enlaces.



The image shows two mobile app screens. The left screen is the 'Login' page, featuring a light blue background with the 'TurismoApp' logo at the top. Below the logo are two white input fields labeled 'Usuario' and 'Contraseña', followed by a blue 'Login' button. At the bottom, there is a link that says 'No tienes cuenta? [Regístrate](#)'. The right screen is the 'Registro' (Registration) page, also with a light blue background. It has a back arrow at the top left. Below the title, there are five white input fields labeled 'Nick', 'Contraseña', 'Repetir Contraseña', 'Nombre', and 'Ciudad'. A blue 'Regístrate' button is positioned below these fields.

Figura 6: Boceto Login y Registro

Cuando accedemos a la aplicación lo primero con lo que nos encontramos es con la página de Login. Desde esta página podemos acceder a la vista Registro si pulsamos el enlace de la parte inferior de la pantalla, o a la aplicación directamente si realizamos el login de manera correcta.



The image shows two mobile app screens. The left screen is a map view of San Francisco, with a blue circle highlighting a specific area. The right screen is the 'Añadir Lugar' (Add Location) form, which has a light blue background. It contains several input fields: 'Titulo', 'Descripción', 'Tipo' (with a dropdown arrow), and 'Ciudad'. Below these is a section for 'Cargar Imagen' (Upload Image) with a 'Cargar' button and a placeholder image. There is also a 'Mapa' button for 'Coordenadas' (Coordinates). At the bottom of the form is a large blue 'Añadir Lugar' button. Both screens have a bottom navigation bar with icons for 'Mapa', 'Añadir Lugar', 'Foro', and 'Perfil'.

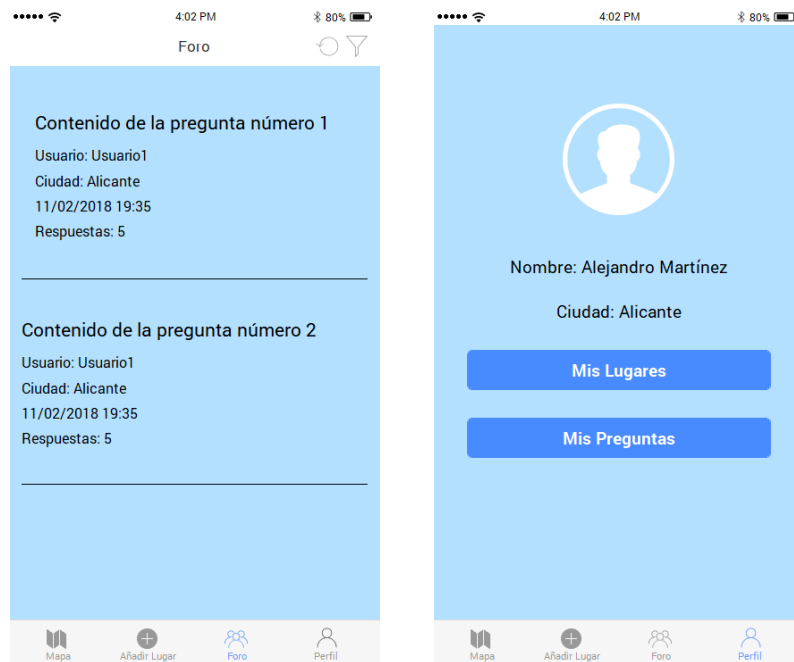
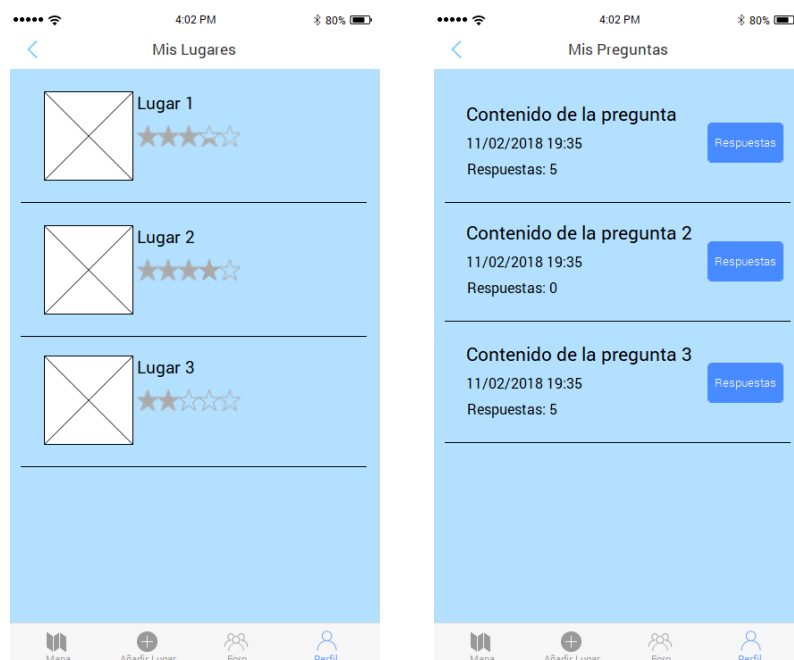


Figura 7: Bocetos vistas principales

Al realizar el login pasamos automáticamente a la página del mapa, el primero de los cuatro apartados principales que conforman la aplicación (Mapa, Añadir lugar, Foro y Perfil). Estos apartados son los que forman el menú inferior, que será visible todo el rato que permanezcamos utilizando la aplicación.



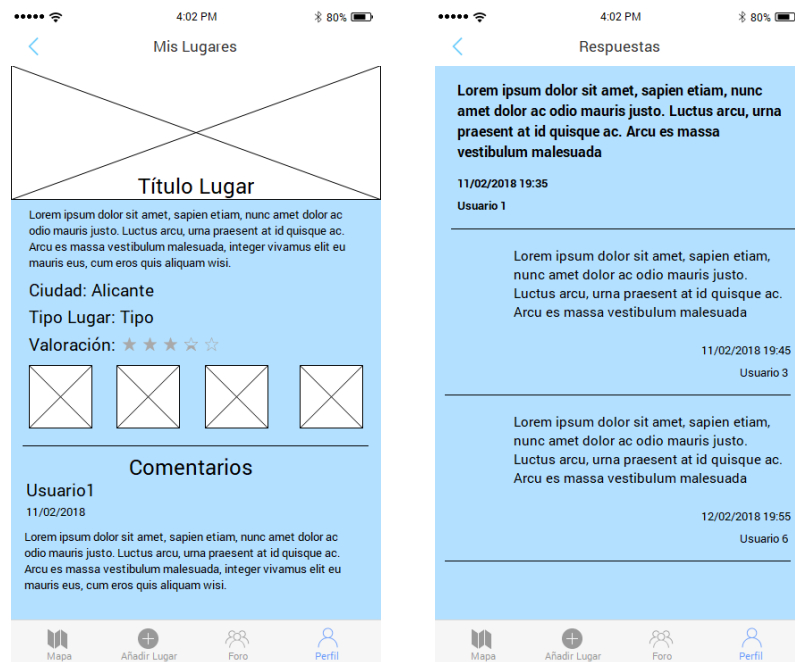


Figura 8: Bocetos vistas secundarias

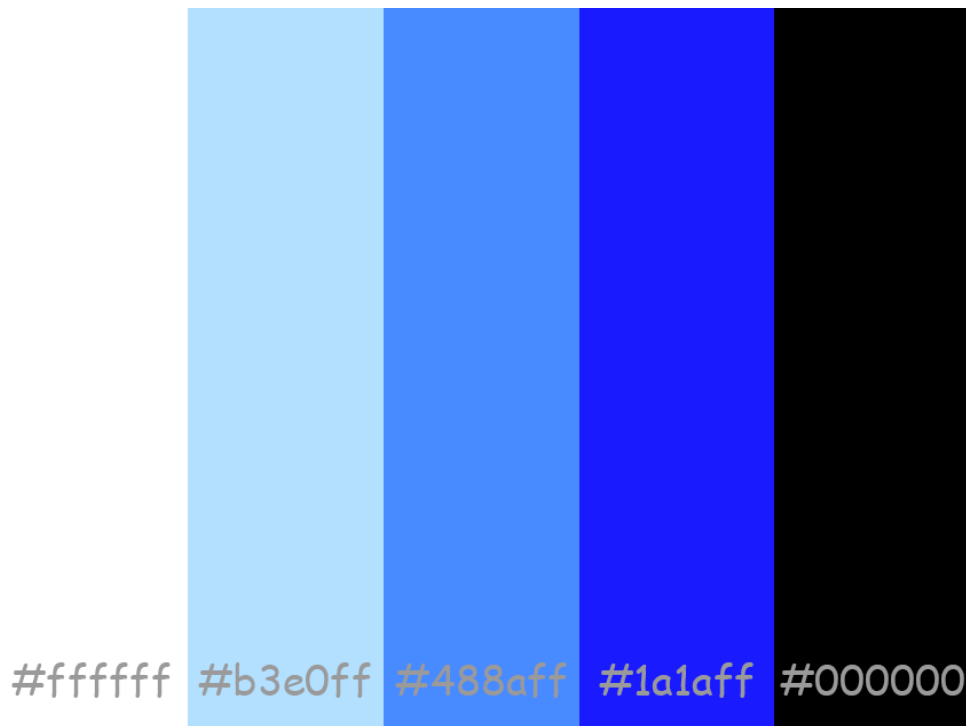
Por último, contamos con las vistas que se podrían definir como de “tercer nivel”. Son las que nos permiten interactuar de manera directa con la aplicación, ya sea viendo las preguntas que hemos realizado en el foro y las respuestas a las mismas, ver la lista de lugares que hemos creado, informarse sobre un lugar y votarlo, o dejar un comentario sobre él.

## 6.5. Guía de estilos

Las distintas interfaces de usuario que conforman la aplicación deben seguir una guía estructurada para no cometer errores a la hora de realizar el diseño final. Para evitar posibles fallos se han confeccionado diferentes guías de estilos a seguir durante todo el proceso de diseño.

### Colores

Al ser una aplicación móvil enfocada al turismo. La elección de los colores es algo a tener en cuenta, ya que muchos de los usuarios decidirán si volver a utilizar la aplicación o no dependiendo de si les atrae lo que están viendo. Para controlar esto se ha confeccionado una paleta de colores a seguir.



*Figura 9: Paleta de Colores Interfaces*

Además de los colores para las vistas, la aplicación cuenta con una serie de marcadores en el mapa que tendrán un color diferente dependiendo del tipo de lugar que estén marcando:

- Parajes naturales – Verde.
- Monumentos y lugares históricos – Marrón.
- Museos – Violeta.
- Construcciones y estructuras – Azul Marino.
- Miradores – Azul.
- Restaurantes – Rojo.
- Otros – Negro.

### **Tipografía**

Se han utilizado diferentes fuentes y estilos de las mismas para diferenciar las diferentes partes del texto en la aplicación.

- El logo consiste en la palabra “AppTurismo” escrita con la fuente “Comic Sans”.
- Todo el texto de la aplicación utiliza la fuente “Roboto” utilizando diferentes variantes en algunos sitios, como la “Roboto bold” para los botones o texto resaltado, además del cambio de tamaño para mostrar los nombres de los sitios o los diferentes apartados de una vista.

## 7. Implementación

Una vez se han especificado los distintos requisitos de la aplicación y se han determinados los factores de diseño pertinentes se da paso a especificar la implementación de los diferentes elementos que conforman la aplicación.

### 7.1. Base de datos

La base de datos de la aplicación (ver Figura 5: Esquema de la base de datos) se diseña y crea en MySQL con la ayuda de MySQL Workbench que es una herramienta visual de diseño de base de datos. Esto nos permite tanto crear la base de datos desde cero, como unir las tablas, añadir o eliminar datos y hacer consultas.

### 7.2. Servidor

Como hemos visto a la hora de investigar sobre las diferentes tecnologías para desarrollar la aplicación, existen varias opciones para programar la capa del servidor.

Se termina por elegir NodeJS ya que cuenta con su propio servidor y es más rápido a la hora de enviar respuestas simultáneas. Además de esto, también es una herramienta cada vez más demandada en el sector del desarrollo web y es ideal para aplicaciones que hacen uso de AJAX.

La estructura del servidor se divide en tres carpetas con diferente información en cada una de ellas. Estas carpetas son las siguientes:

#### **Config**

Esta carpeta es la que se encarga de guardar la información de configuración de diferentes puntos del servidor. En concreto cuenta con dos archivos:

- Auth.js: Este archivo es donde se guarda la clave secreta de los tokens que se utilizan en la autenticación (de la que se hablará más adelante).
- Configbd.js: Aquí están almacenados los datos y variables de configuración para conectar con la base de datos.

Estos archivos están divididos del resto por temas de simplicidad y seguridad, ya que no es conveniente que en cada archivo que usemos token o hagamos llamadas a las bases de datos tengamos que escribir la información al inicio de cada documento.

### **Models**

En esta carpeta tenemos tres archivos (places, questions, users) y es donde se crean las funciones que tienen en su interior las distintas consultas SQLs para hacer las llamadas a la base de datos y obtener los datos pertinentes. Cada uno de estos tres archivos tiene las funciones necesarias para obtener de la base de datos la información que los usuarios piden desde la aplicación (ver Tabla 3: API Rest).

### **Routes**

Es la carpeta más importante, contiene 4 archivos, tres que son iguales a los mencionados en models (places, questions, users) y uno llamado index.js.

Estos archivos cuentan con todas las rutas de la API (ver Tabla 3: API Rest). Cada uno de los archivos cuenta con sus propias rutas para que el código sea mucho más claro. Index.js cuenta sólo con dos rutas, una para crear un usuario y otra para hacer el login. Esto es así porque este archivo es el único de los cuatro que no es de carácter privado, lo que significa que cualquier usuario puede acceder a estas dos rutas sin necesidad de tokens.

Los otros tres archivos cuentan con una función al inicio de cada uno de ellos para comprobar si les entra o no un token y si es correcto. En caso de ser incorrecto la petición se anula y devuelve un mensaje de error al usuario.

## **7.3. Seguridad en el servidor**

La seguridad en los servidores es cada vez más importante, ya que se realizan ataques, o se buscan puntos ciegos para poder acceder y obtener información delicada. Por ello siempre hay que tomar ciertas medidas de seguridad para evitar que esto suceda, y brindar al usuario la seguridad de que nadie va a poder obtener o manipular su información privada.

### **Cifrado de las contraseñas**

Para el cifrado de contraseñas de los usuarios se ha utilizado la librería bcrypt de NodeJS. Esto nos permite cifrar las contraseñas para que, si alguien obtiene la información de nuestra base de datos, no pueda ver directamente la contraseña del usuario en cuestión.

En concreto utilizamos un hash que está compuesto por la contraseña cifrada del usuario y de una clave secreta almacenada en el servidor.

### **Inyección de código SQL**

Desde hace un tiempo se ha extendido el uso de las inyecciones SQL gracias a que es una manera fácil de obtener datos directamente de la base de datos utilizando sólo una cadena de caracteres adecuada.

Evitamos que esto pueda ocurrir gracias al uso de la función 'connection.escape()' que nos proporciona Express y que garantiza que no se puedan realizar inyecciones de código al escribir la URL y enviarle la petición al servidor:

```
"SELECT * FROM lugares where tipo="+connection.escape(tipo)
```

### **JSON web token (JWT)**

Para la autenticación de los usuarios al entrar en la aplicación se ha decidido hacer uso de tokens ya que es una forma de transmitir información y privilegios sin utilizar mucho espacio y que nos ahorra realizar peticiones innecesarias a la base de datos, ya que contiene la información que se crea conveniente.

La manera en la que nuestra aplicación utiliza los tokens es la siguiente:

Hay dos maneras de crear un token, mediante el registro de un usuario nuevo o al hacer login. Cuando realizamos una de estas dos acciones y el backend comprueba que los datos introducidos son correctos, se crea un token utilizando la información del usuario (en concreto el Nick y la contraseña). Esta información se envía al frontend de manera que lo guardamos en la aplicación y se adjunta como cabecera cada vez que realizamos una petición a la API que necesite de token.

Para la aplicación hemos hecho uso de una librería de NodeJS llamada JSON Web Token. Esto genera un token con los datos del usuario que hemos elegido y una clave secreta, aplicando el algoritmo MD5 con una caducidad de 24 horas (caducidad variable). Una vez hecho esto le pasamos el token desde el backend a la aplicación y lo almacenamos en el localStorage (almacenamiento local de cada aplicación) mediante un plugin que nos proporciona Ionic.

```
1      router.use(function(req, res, next) {  
2          // buscamos el token en el header
```

```

3      var token = req.headers['token-acceso'];
4      // lo decodificamos
5      if (token) {
6          // verificamos la clave secreta
7          jwt.verify(token,configJWT.secret,          function(err,
decoded) {
8              if (err) {
9                  return res.json({ success: false, message: 'Failed to
authenticate token.' });
10             } else {
11                 // si todo es correcto, nos permite realizar peticiones en
el documento
12                 req.decoded = decoded;
13                 next();
14             }
15         });
16     } else {
17         // si no obtenemos token, nos devuelve un error
18         res.json(200,"No token provided.");
19     }
20 });

```

## Seguridad de datos delicados

Otro de los puntos a tener en cuenta para desarrollar una buena seguridad en el proyecto es no tener a la vista información delicada, en este caso hablamos de los datos relacionados con la conexión de la base de datos y de la clave secreta utilizada para generar tokens.

Para evitar que estén a la vista en cualquier archivo de la API, lo que se ha realizado es un documento auxiliar con estos datos, al que se llama desde el resto de archivos que precisen esta información. Para ello hacemos una llamada a ese archivo que nos devuelve los datos necesarios sin necesidad de que aparezcan en el archivo de destino.

```

1      connection=mysql.createConnection({
2      host: configDB.dbreserved.host,
3      user: configDB.dbreserved.user,
4      password: configDB.dbreserved.password,
5      database: configDB.dbreserved.database
6      });

```



## 7.4. Cliente

La parte del cliente de la aplicación se ha desarrollado utilizando la herramienta Ionic, más concretamente Ionic3.

Ionic es una herramienta gratuita open source que sirve para crear aplicaciones híbridas hechas mediante HTML5 CSS y JavaScript. Como se ha dicho anteriormente, para este proyecto se ha utilizado Ionic3 que trae una serie de novedades como son:

- Angular 4.0
- TypeScript 2.1
- IonicPage Decorator

Con respecto a la estructuración del cliente es bastante sencilla. Ionic nos proporciona una serie de facilidades a la hora de realizar el trabajo. También cabe mencionar que todos los componentes que se van a definir a continuación son creados mediante el comando “Ionic Generate”.

### Pages

Esta carpeta contiene todas las páginas o vistas que conforman la aplicación. Cada una de las páginas cuenta con cuatro archivos:

- Un archivo HTML donde se encuentra todo el código necesario para mostrar y dividir la información, es decir, es el template de la página.
- Un archivo SCSS que es en el que definimos los estilos que queremos darle a esa página en concreto.
- Un archivo MODULE.TS, donde se define el módulo de la página.
- Por último, un archivo .TS que es donde se encuentra toda la lógica de la página.

### Providers

Esta aplicación también cuenta con **providers**. Los Providers son los archivos encargados de manipular los datos externos como pueden ser conexiones con una API REST, conectar con localStorage para obtener información...

En concreto, la aplicación cuenta con un solo provider llamado users que es el encargado de hacer todas las peticiones a la API, obtener la información y enviársela a las páginas que la necesiten.

## Assets

Esta carpeta sirve para guardar y proporcionar a las páginas o los estilos las imágenes necesarias. En este caso la carpeta cuenta con una serie de iconos que son utilizados para marcar en el mapa los diferentes puntos de interés y diferentes imágenes como son el logo o la imagen predeterminada que aparece en el área de perfil de los usuarios.

## Otros archivos

Además de las carpetas mencionadas con anterioridad, el proyecto cuenta con otros archivos que también son de interés:

- `App.module.ts`: Este archivo es el encargado de inicializar todos los componentes o plugins que van a ser utilizados en la aplicación: páginas, providers, geolocalización, cámara del dispositivo...
- `App.scss`: Archivo en el que se definen los estilos globales. Sirve para declarar un estilo que va a ser aplicado en todas las páginas de nuestra aplicación.
- `Index.html`: En este html es donde se inicializan los scripts que, posteriormente, serán utilizados en el resto de la aplicación, por ejemplo, la llamada al mapa que nos proporciona Google Maps.

## 7.5. Manipulación DOM

Al ser una aplicación turística que cuenta con diferentes mapas con información para los usuarios nos encontramos ciertos problemas a la hora de manipular y mostrar la información. Uno de los principales problemas nos lo encontramos al manipular el DOM.

El servicio de Informática de la Universidad de Alicante define el DOM (Document Object Model) como “una interfaz de programación para los documentos HTML y XML. Facilita una representación estructurada del documento y define cómo y de qué manera pueden acceder los programas a la estructura, estilo y contenido del mismo. El DOM es una representación del documento como un grupo de nodos y objetos estructurados con propiedades y métodos propios”.

El problema viene cuando, una vez ya creado y estructurado este documento, intentamos modificarlo desde Angular (la aplicación esta realizada con Ionic, el cual funciona sobre AngularJS).

Cuando modificamos directamente desde el controlador el DOM se crea un acoplamiento indeseado entre dos capas, la de la lógica y la de renderizado. Este acoplamiento nos impide, entre otras cosas, lanzar la aplicación de determinadas maneras. Esto ocurre porque Angular es una *platform agnostic*, lo que quiere decir que está diseñada de manera que separa toda su lógica y funcionamiento del renderizado del DOM.

Para evitar todo esto Angular, y por consiguiente Ionic, cuenta con una clase llamada `Renderer` (ya obsoleta, ahora la clase utilizada se llama `Renderer2`) que nos ayuda a manipular el DOM sin afectar al encapsulamiento de los componentes de Angular.

El propio Angular considera un mal uso la manipulación directa del DOM, por ello en esta aplicación se utilizan diferentes clases (`Renderer2`) para evitar los acoplamientos anteriormente mencionados porque es necesario manipularlo, ya que mostramos información ajena a Google Maps que sólo contiene la base de datos de la aplicación y para poder interactuar con esta información, por ejemplo, seleccionando uno de los puntos de interés personalizados y mostrando su información detallada, o la ruta para llegar a este lugar desde nuestra posición.

## 7.6. Tratamiento de imágenes

Como en la mayoría de aplicaciones de índole turístico, esta aplicación cuenta con un factor importante, tanto estético como informativo: las imágenes.

A la hora de crear una aplicación móvil que cuenta con una cantidad razonable de imágenes es casi obligatorio crear o utilizar algún tipo de compresión en las mismas. En este caso, cada punto de interés creado por la comunidad contará, como mínimo, con una imagen principal, y si el usuario lo desea, podrá añadir 3 imágenes más. Por tanto, para no hacer uso de una cantidad superior a la necesaria de datos móviles y para agilizar la navegación entre páginas se ha utilizado una funcionalidad incluida en el plugin de Cordova, que es utilizado para cargar las imágenes tanto de la galería del dispositivo como de la propia cámara. Este plugin nos permite comprimir las imágenes hasta el punto que nosotros creamos necesario, reduciendo de manera significativa el tamaño de los archivos.

Esta compresión la logramos cambiando el parámetro `Quality` dentro de las opciones que configuramos cuando hemos elegido la imagen a subir (ya sea desde la cámara o desde la galería).

Gracias a esta opción conseguimos que, por ejemplo, una imagen que ocupa 14,49 MB en el dispositivo pase a ocupar 52,6KB en el servidor que aloja nuestra API. Todo esto conservando un mínimo de calidad para que las imágenes no pierdan casi detalle (todo partiendo del hecho de que se abren desde pantallas reducidas de dispositivos móviles).

## 8. Pruebas y validación

### 8.1. Validación de resultados

Una vez terminado el desarrollo de la aplicación el siguiente paso es la comprobación de que todo funciona correctamente, siguiendo los requisitos establecidos con anterioridad y sin pequeños errores puntuales. Para verificar que todo funciona según lo establecido se ha hecho una serie de pruebas en diferentes puntos.

#### **Login y Registro**

En estas pruebas se va a comprobar todo lo relacionado con la gestión de los usuarios de nuestro sistema. Las comprobaciones llevadas a cabo son las siguientes:

- Registro: Un usuario se puede registrar en la aplicación y acto seguido, su información es guardada en la base de datos con su contraseña encriptada para evitar posibles problemas.
- Registro de un usuario duplicado: Si al registrarnos utilizamos un nombre de usuario ya existente en el sistema la aplicación nos muestra una alerta avisando de que ese nombre no está disponible.
- Confirmar contraseña: Cuando nos registramos debemos introducir la contraseña dos veces para evitar posibles equivocaciones y registrarnos con una contraseña que luego no podamos recordar.
- Login: Un usuario ya registrado puede acceder a la aplicación con su nombre y contraseña en cualquier momento.
- Comprobación Login: Cuando introducimos algún dato erróneo al iniciar sesión, el sistema nos muestra un aviso de que el nombre de usuario o la contraseña no son correctas.

Todos los puntos anteriores han sido probados y funcionan correctamente.

#### **Mapas**

La interacción con los mapas es uno de los puntos más importantes de la aplicación, ya que tenemos que comprobar que funcionen correctamente en los diferentes casos que se nos presentan.

La API de Google Maps sólo nos permite cargar un único mapa en nuestra aplicación, por lo que tenemos que ir variando la información que se muestra y la manera de interactuar con él. Los 3 casos que deben funcionar son los siguientes (Figura 10: Mapas de la Aplicación):

- Mapa con nuestra posición actual y los puntos de interés cercanos: Al iniciar sesión lo primero que vemos es la pantalla principal del mapa, donde nos aparece un puntero marcando la posición en la que nos encontramos y los diferentes lugares de interés que existen en la zona.
- Mapa de ruta: Cuando elegimos lugar de inicio y fin o pulsamos el botón de “Cómo llegar” a un lugar de interés, el mapa cambia para mostrarnos, en ese espacio, un mapa más pequeño con la ruta a seguir y, a continuación, los diferentes pasos a seguir para llegar a nuestro destino.
- Marcar posición de lugar de interés: El último tipo de mapa aparece completamente vacío y nos permite pulsar en un lugar del mismo para indicar dónde se encuentra el lugar que vamos a crear.

Todos los casos citados anteriormente funcionan de forma satisfactoria. Además, se ha comprobado que el código correspondiente para marcar la posición actual del usuario mediante GPS funcione como es debido.

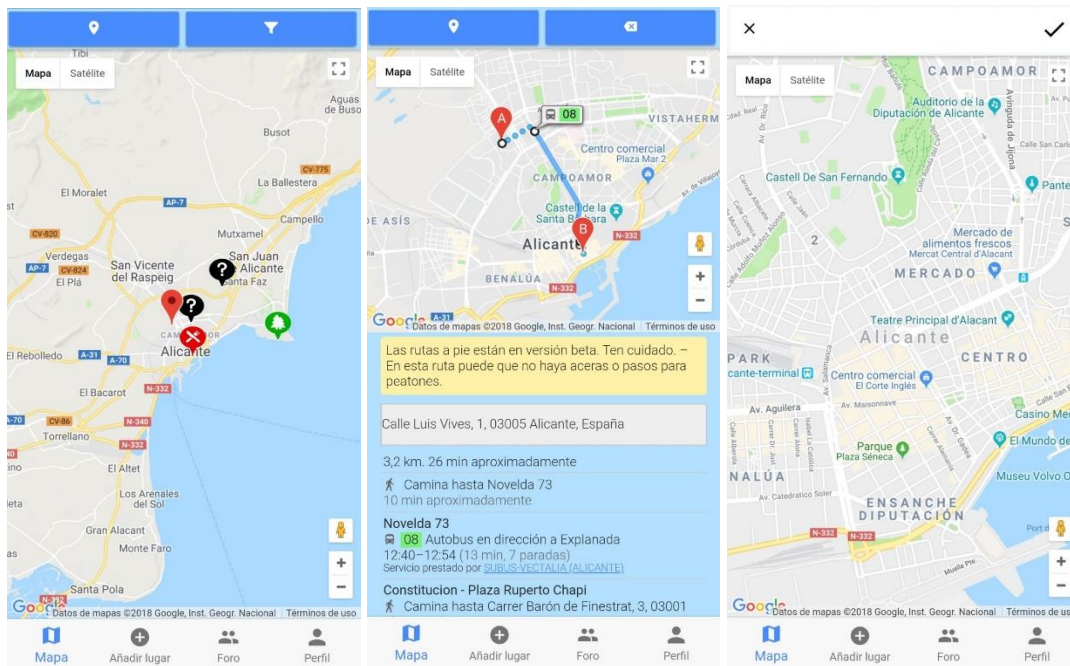


Figura 10: Mapas de la Aplicación

## Registro de lugares

También se ha comprobado que, a la hora de crear un lugar de interés, todo funcione como es debido. Para ello se han insertado en la base de datos varios lugares haciendo diferentes pruebas.

Cuando creamos un lugar es obligatorio rellenar todos los campos, además, si no abrimos el mapa y marcamos el lugar donde se encuentra el sitio, la aplicación nos avisa de que no hemos

marcado un lugar en el mapa y no nos deja registrar la información. Aparte, también es obligatorio añadir una imagen principal al sitio y, cuando toda la información es correcta y se crea el lugar, nos aparece una ventana que nos da la opción de añadir más fotos para así proporcionar más datos al resto de usuarios. Si accedemos, se nos abre otra página en la que se pueden cargar y previsualizar 3 fotos secundarias.

## 8.2. Pruebas y resultados de Usabilidad

Como ya se ha dicho varias veces, al crear una aplicación de turismo se debe comprobar que se han realizado unas interfaces usables para todo tipo de usuarios, ya sean usuarios experimentados o usuarios que no estén muy familiarizados con la tecnología.

La manera elegida para realizar las comprobaciones de la usabilidad de esta aplicación ha sido, al terminar todo el desarrollo de la aplicación, dejar que distintos tipos de usuarios la utilicen durante un tiempo para así obtener diferente feedback.

El primer tipo de usuario al que se le ha proporcionado la posibilidad de testear la aplicación han sido familiares que no tienen mucha experiencia en el uso de tecnologías. Estos usuarios han permitido corregir errores menores en la maquetación y distribución de la información como son:

- Elegir un color diferente para cada tipo de lugar de interés marcado en el mapa.
- Añadir una imagen de fondo en la pantalla de Login.

El segundo tipo de usuario ha sido otro familiar que interactúa y convive a diario con dispositivos móviles, pero sin ningún conocimiento relacionado con la programación o creación de aplicaciones. Esta prueba ha servido para corroborar que las mejoras propuestas por el primer grupo de usuarios era necesario llevarlas a cabo.

El tercer y último grupo en testear la aplicación ha sido una serie de compañeros de Ingeniería Multimedia. Este grupo es el que más mejoras ha propuesto para la aplicación ya que tiene una serie de conocimientos avanzados sobre programación. Algunas de las propuestas han sido:

- Tiempos de carga.
- Errores al mostrar imágenes.
- Interacción con los mapas

Una vez terminada la fase de testeo de la aplicación, se estudiaron los resultados y se llegó a la conclusión de que había que realizar algunos pequeños cambios para mejorar el producto final. Estos cambios son los siguientes:

- Se ha incluido una imagen con un pequeño desenfoque para el fondo de la pantalla de login, en lugar de un color base.
- Se han cambiado los colores de los marcadores para los lugares de interés dependiendo de la categoría que les corresponda.
- Se ha añadido una pequeña mejora a la compresión de imágenes para que no aparezcan problemas al mostrarlas si se accede muy rápido a una interfaz.
- Se han incrementado levemente el tiempo de los *Loading* de inicio de sesión y acceso a la información de un lugar para permitir que cargue toda la información correctamente.



## 9. Resultados

### 9.1. Aplicación

Como resultado final se ha obtenido una serie de interfaces que componen la aplicación y hacen que funcione adecuadamente. A continuación, vamos a definirlas en profundidad.

#### 9.1.1. Login y Registro

La página de login es la primera que nos encontramos al abrir la aplicación. Está compuesta por el logo de la aplicación, un pequeño formulario para introducir nuestro nombre de usuario y contraseña y un botón para iniciar sesión. Además de esto cuenta con un enlace a la página de registro para los nuevos usuarios.

Por otra parte, tenemos la página de registro. Esta página ya sigue la dinámica de toda la aplicación (colores azules para el fondo, botón y formulario como en el resto de interfaces etc.)

El único cambio con respecto a los mockups iniciales es la sustitución de un color plano para el fondo de la página de login por una imagen con un pequeño desenfoque gaussiano que le da una apariencia más llamativa.

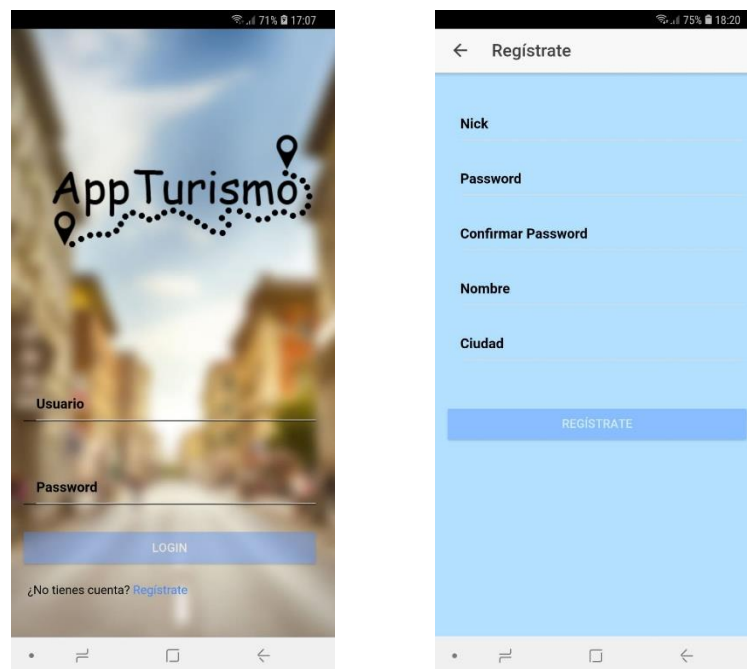


Figura 11: Páginas Login y registro

### 9.1.2. Mapa

Esta es la página que vemos cuando realizamos el login o el registro de manera correcta. Esta interfaz cuenta con únicamente el mapa (centrado en nuestra posición actual) y dos botones.

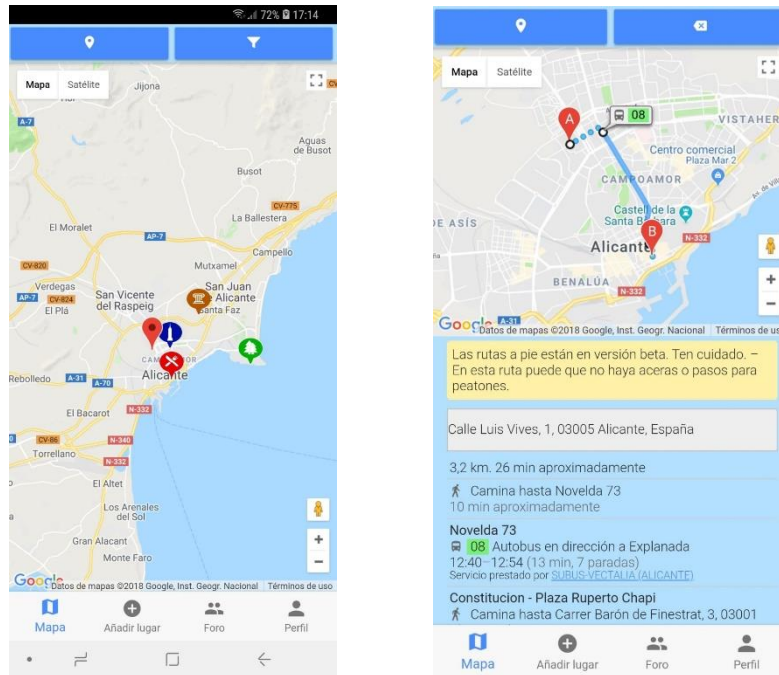


Figura 12: Página de Mapa

En el mapa podemos observar nuestra posición actual y unos marcadores que corresponden con los lugares de interés creados por la comunidad. Al pulsar en alguno de estos lugares se nos despliega una ventana que contiene el nombre, foto y puntuación del lugar, además de dos enlaces que nos permiten abrir la información detallada del sitio y otro que nos traza una ruta hasta este punto desde nuestra ubicación actual.

Con respecto a los dos botones de la parte superior, estos sirven para desplegar dos formularios:

- El primer formulario nos permite trazar una ruta desde un lugar de inicio a uno de destino, introducidos manualmente. La aplicación cuenta con un conjunto de funciones que nos muestran una serie de opciones para autocompletar antes de tener que rellenar todo el campo de texto. Una vez elegidos los dos puntos de la ruta, podemos escoger la manera en la que nos queremos desplazar (a pie, en coche o en transporte público). Una vez hecho esto, la aplicación calcula la ruta y la página del

mapa cambia para mostrar un mapa más pequeño con la ruta y una serie de indicaciones para llegar a nuestro destino.

- El segundo botón abre un filtro en el que podemos elegir qué tipo de lugar se desea ver, por ejemplo, si se elige “restaurantes” desaparecerán todos los lugares de interés menos los etiquetados como restaurantes.

### 9.1.3. Añadir Lugar

La siguiente página, la segunda que aparece en el menú inferior con el que cuenta la aplicación, es un sencillo formulario que permite crear un lugar de interés. Es obligatorio rellenar todos los campos, añadir una foto (que podremos visualizar en la parte derecha de la pantalla) y añadir una ubicación, para ello hay que acceder al mapa mediante el botón “Ver Mapa” seleccionar un lugar y marcar el *tick* que aparecerá en la parte superior derecha.

Hecho esto nos sale un mensaje confirmando que hemos creado el lugar y nos da la opción de subir 3 imágenes más siguiendo el mismo método que antes.

También cabe mencionar que contamos con la posibilidad de subir fotos tanto desde la galería del dispositivo como abriendo la cámara y tomando una foto en el acto.

Figura 13: Página Añadir Lugar

#### 9.1.4. Páginas foro y respuestas

La siguiente parte de nuestra aplicación es la relacionada con el foro. Todo usuario dado de alta en la plataforma puede crear una pregunta para el foro y también puede responder las preguntas de otros usuarios. Al entrar en la página del foro se muestra una lista de preguntas, ordenadas por novedades. Esta opción se puede cambiar con el icono que hay arriba a la derecha, que permite filtrar las preguntas dependiendo de la ciudad a la que van dirigidas. Se puede borrar el filtro en cualquier momento para volver a visionar todas las preguntas.

Se dispone de un conjunto de información general sobre las preguntas tales como la fecha, el título o el número de respuestas que han recibido. Si queremos acceder a las respuestas sólo tenemos que pulsar sobre la pregunta y directamente nos enviará a la página de esa pregunta en concreto.

Esta página de respuestas es bastante simple, sólo cuenta con la pregunta resaltada y las respuestas que se han dado a la misma.

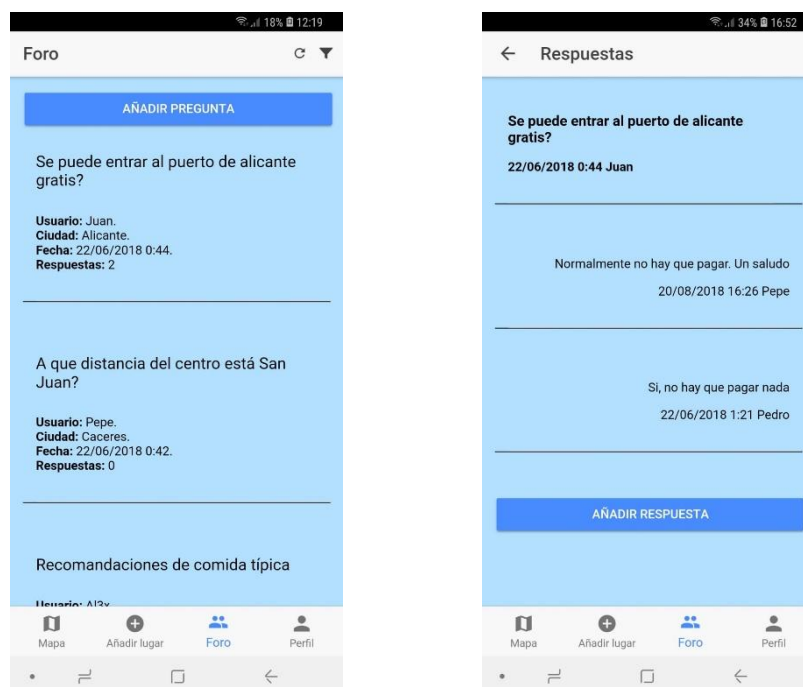


Figura 14: Página foro y respuestas

#### 9.1.5. Añadir pregunta

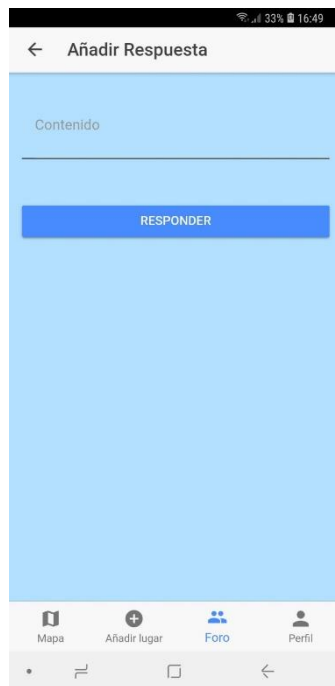
Cuando pulsamos el botón de “Añadir pregunta” en el foro nos envía a esta página. Es un pequeño formulario en el que se rellena el contenido de la pregunta que se quiere enviar al foro y la ciudad

donde va destinada. Una vez hecho esto nos aparece un mensaje avisando de que la pregunta se ha creado correctamente y acto seguido nos redirecciona de nuevo a la página del foro.

*Figura 15: Página añadir lugar*

#### 9.1.6. Añadir Respuesta

A este formulario se accede cuando, dentro de la página que muestra una pregunta junto a sus respuestas, pulsamos sobre el botón “Añadir respuesta”. Es simple, sólo cuenta con un cuadro a rellenar donde se introduce el contenido de la respuesta que queremos enviar y un botón el cual, si hemos rellenado el cuadro de contenido, nos mandará a la página principal del foro junto con un mensaje confirmando que hemos enviado la respuesta a la pregunta de manera correcta.



*Figura 16: Página añadir respuesta*

#### 9.1.7. Perfil

Última página que conforma el menú de la aplicación. Aquí podemos observar la información del usuario que ha iniciado sesión, además de dos botones:

- El botón de “Mis lugares” redirige a una página con la lista de los lugares de interés que ha creado el usuario logueado. Aparece de cada uno el nombre, puntuación y foto principal. Si pulsamos sobre el lugar se muestra toda su información detallada.
- El botón de “Mis preguntas” te muestra una página con la lista de preguntas que el usuario ha lanzado al foro. Se puede ver cuantas respuestas has recibido en la pregunta y acceder a sus respuestas. Esta página de respuestas es la misma que la que aparece en la (Figura 14).

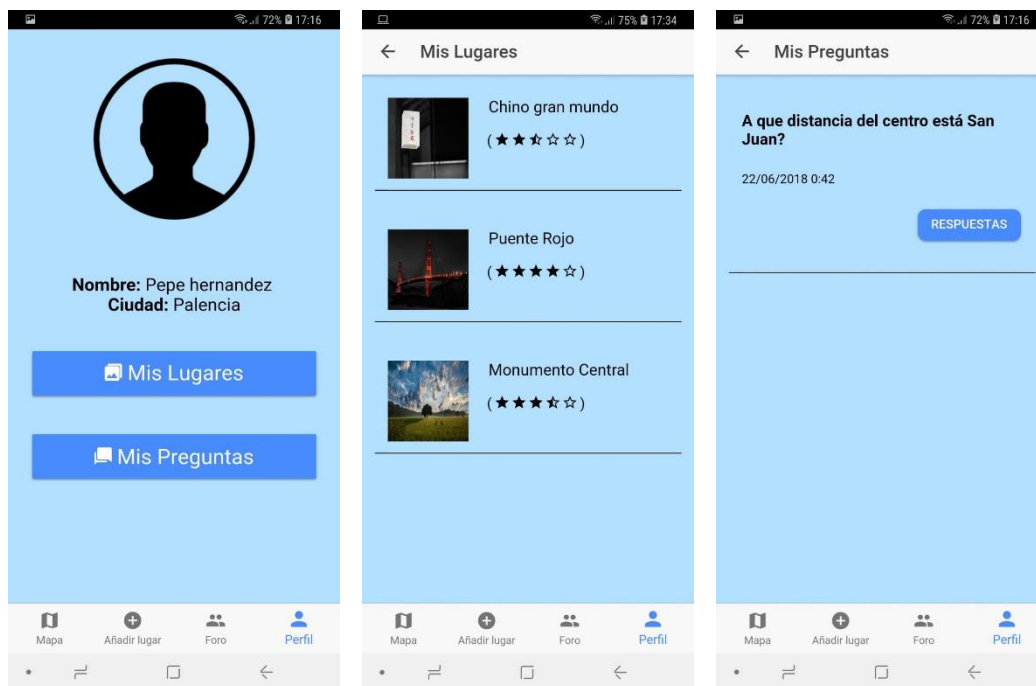


Figura 17: Páginas Perfil, Mis lugares, Mis respuestas

#### 9.1.8. Información de un lugar

Esta página es la que muestra la información detallada de un lugar. Aparece toda su información, las imágenes, tanto principal como secundarias, los comentarios que ha recibido y nos da la posibilidad de comentar sobre este lugar o votarlo. Además de esto tenemos la opción de pulsar sobre las imágenes para ampliarlas y verlas con más detalle.

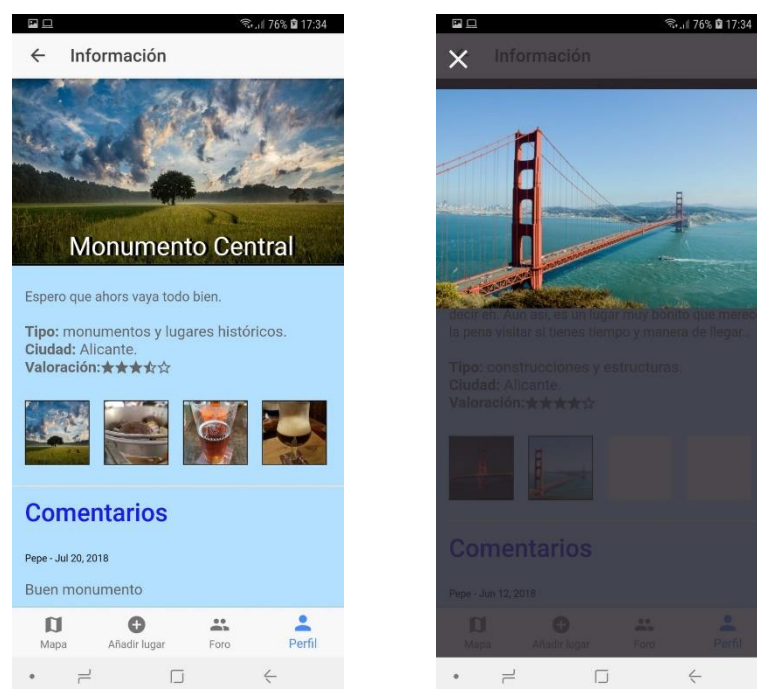


Figura 18: Página Información Lugar y vista previa

## 10. Conclusiones

Tras terminar el desarrollo de este trabajo se puede observar cómo se han cumplido los objetivos contemplados en el producto que se deseaba conseguir. Se puede ver como se ha conseguido crear y mantener un servidor que aloja una API REST funcional, con una base de datos sólida y una aplicación atractiva y con todas las funcionalidades esperadas al inicio del proyecto.

Aún con los resultados visibles, la aplicación puede generar algo más que las funcionalidades empleadas. Se puede conseguir un producto mejorado en algunos aspectos y de mayor envergadura ya que, como se contempla en el modelo de negocio (Figura 2: Lean Canvas), uno de los propósitos es la distribución del producto.

Para ello se han estudiado una serie de mejoras basándose en la investigación y el feedback proporcionado por los usuarios que han podido probar la aplicación.

### 10.1. Trabajo futuro

Lo que se buscaba a la hora de llevar a cabo la idea de este proyecto es obtener un producto funcional que brindase una ayuda a determinados usuarios. Con ese objetivo cumplido, con el paso del tiempo han ido surgiendo una serie de mejoras o ampliaciones que harían a la aplicación llegar a un mayor número de usuarios o conseguir mejores valoraciones por parte de la comunidad. Con estas mejoras se conseguiría que la distribución del producto fuese algo más viable y real. Para ello se ha hecho una lista con las principales funcionalidades o mejoras a contemplar:

- Mejora del perfil del usuario: En estos momentos el apartado donde los usuarios pueden ver su información personal es bastante simple, contando únicamente con los datos principales más utilizados. Una de las posibles mejoras sería la ampliación de este apartado, añadiendo una foto para cada usuario y una puntuación basándose en la popularidad de sus lugares creados o sus respuestas en el foro etc.
- Rutas personalizadas: Una de las primeras mejoras que se propusieron al ver el producto final fue la creación de un apartado que permitiese a los usuarios crear rutas que mostrasen diferentes lugares de interés al resto de usuarios. Estas rutas tendrían la misma dinámica que los lugares, ya que contarían con sus comentarios, sus valoraciones etc.



- Sugerencias de lugares: Todos tenemos preferencias a la hora de visitar ciertos lugares, para ello una buena mejora sería poder recoger la información de los lugares que ya has visitado para poder ofrecerte una lista de los lugares que aún no has visitado y pueden ser de tu interés.

Las mejoras anteriores son de cara a los usuarios que van a utilizar la aplicación, pero también hay que tener en cuenta que, para mantener una buena aplicación, debemos contar con una buena administración de la misma. Para ello una de las mejoras más grandes y más importantes sería la creación de un panel de administrador para poder gestionar toda la información relacionada con la plataforma. Algunas de las funcionalidades con las que contaría este panel serían las siguientes:

- Gestión de usuarios: Darle al administrador la posibilidad de crear o dar de baja a algún usuario si fuese necesario, o incluso modificar su información.
- Gestión de lugares: Un administrador también tendría la opción de modificar o incluso eliminar algún lugar si la situación lo requiere.
- Cuadros de mando: Una de las cosas más importantes a la hora de comercializar una aplicación es contar con una serie de estadísticas que nos muestren cómo avanza nuestro producto. Para conseguir esto se incluirían diversos cuadros de mando que nos mostrarían información específica, como podría ser usuarios recurrentes, alta de nuevos usuarios y lugares, participación en el foro...

## 11. Bibliografía y referencias

- 1 Adobe. PhoneGap: <http://phonegap.com>
- 2 Angular. Angular 5: <https://angular.io/>
- 3 Express. Express 4.x: <http://expressjs.com/>
- 4 Facebook. React Native: <https://facebook.github.io/react-native/>
- 5 Framework7: <https://framework7.io>
- 6 Google. Obtener clave y autenticación | Google Maps Platform: <https://cloud.google.com/maps-platform/?hl=es>
- 7 Google. Mapas | Maps Javascript API | Google Developers: <https://developers.google.com/maps/documentation/javascript/tutorial>
- 8 Google. Rutas | Google Maps Platform Routes: <https://cloud.google.com/maps-platform/routes/>
- 9 Ionic. Ionic framework: <https://ionicframework.com/>
- 10 Josh Morony. Integrate Google Maps and Geolocation. <https://www.joshmorony.com/>
- 11 Json Web Token. Autenticación: <https://jwt.io/>
- 12 JustInMind. Prototyping tool: <https://www.justinmind.com/>
- 13 Microsoft. Microsoft SQL Server: <https://www.microsoft.com/es-es/sql-server/sql-server-2017>
- 14 Microsoft. Xamarin: <https://visualstudio.microsoft.com/es/xamarin/?rr=https%3A%2F%2Fwww.google.es%2F>
- 15 Mongo. MongoDB: <https://www.mongodb.com/>
- 16 SQL. MySQL: <https://www.mysql.com/>
- 17 Node. NodeJS: <https://nodejs.org/en/>
- 18 NPM. Node Packages: <https://www.npmjs.com/>
- 19 OMT | Organización Mundial del Turismo: <http://www2.unwto.org/es>
- 20 Oracle. Oracle Database: <https://www.oracle.com/es/database/index.html>
- 21 Redis. Redis: <https://redis.io/>
- 22 The Valley Digital Business School: <https://thevalley.es/>
- 23 VertX. Eclipse Vert.x: <https://vertx.io/>
- 24 W3Schools. AngularJs and SQL Tutorials: <https://www.w3schools.com/>

